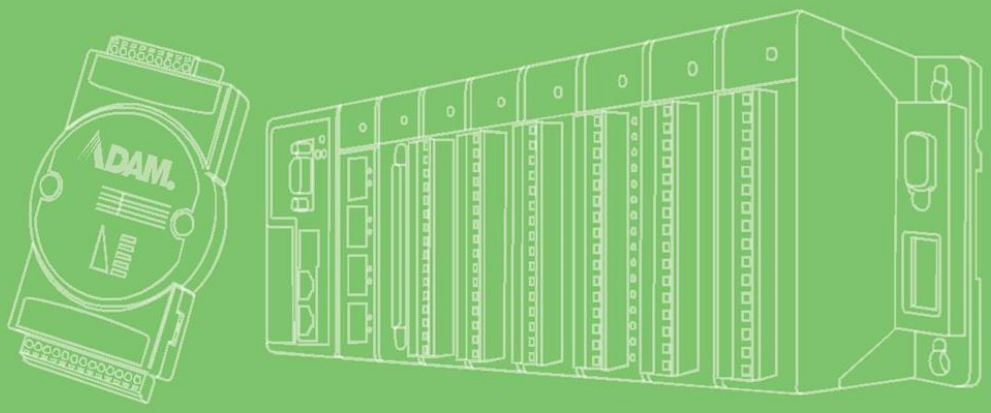


User Manual

Advantech CODESYS for RTE X86

CONTENTS

1. Introduction 8

1.1. About This Manual 8

1.2. Organization of This Manual 8

2. Installations 11

2.1. CODESYS Installation 11

2.2. Add-on Package Installation 15

2.2.1. Installation 15

2.2.2. Updating the Package 17

3. Create and run a project 20

3.1. Start CODESYS 20

3.2. Create a Project 21

3.3. Write a Program 23

3.4. Connect to the Target Device 25

3.5. Run the Application 26

4. System Diagnosis 29

4.1. System Information 29

4.2. Map Variables to System Information 30

5. Advantech I/O Modules 33

5.1. APAX-5000 IO Modules 33

5.1.1. Scan I/O Modules into CODESYS 33

5.1.2. Insert I/O Modules into CODESYS 36

5.1.3. Map Variables to I/O Modules 38

5.1.4. Support List 40

5.1.5. Digital Input Modules 40

5.1.6. Digital Output Modules 43

5.1.7. Analog Input Modules 45

5.1.8. Analog Output Modules 47

5.1.9. Relay Output Modules 49

5.2. AMAX-5000 IO Modules 52

5.2.1. Scan AMAX I/O Modules into CODESYS 52

5.2.2. Insert AMAX I/O Modules into CODESYS 58

5.2.3. Map Variables to I/O Modules 61

5.2.4. Support List 64

5.2.5. EtherCAT Slave Modules Configuration 65

5.2.6. Digital Input Modules 70

5.2.7. Digital Output Modules 72

5.2.8. Analog Input Modules 75

5.2.9. Analog Output Modules 78

5.2.10. Counter Modules 81

5.2.11. Timestamp Modules 84

5.2.11.1 EtherCAT Timestamp Templates 85

5.2.11.2 AMAX-5051T Digital Input with Timestamp 86

5.2.11.3 AMAX-5056T Digital Output with Timestamp 92

5.2.11.4 EtherCAT Distributed Clocks Synchronization 93

6. Advantech Fieldbus Modules 96

6.1. Modbus 96

6.1.1. Modbus RTU Client 96

6.1.1.1. ADAM-4000 Series 104

6.1.1.1.1. Read Coil Status 105

6.1.1.1.2. Read Holding Registers 106

6.1.1.1.3. Force Multiple Coils 107

6.1.1.1.4. Preset Multiple Registers 109

6.1.1.2. ADAM-5000 Series 110

6.1.2. Modbus TCP Client 113

6.1.2.1. ADAM-6000 Series 117

6.1.2.2. ADAM-5000 Series 119

6.1.3. Modbus TCP Server 120

6.2. CANOpen 124

6.2.1. CANOpen Client 124

6.2.1.1. Configuration Files Installation 124

6.2.1.2. Scan for Slaves 125

6.3. EtherNet/IP 132

6.3.1. EtherNet/IP Client 132

6.3.1.2. Configuration Files Installation 132

6.3.1.3. Add Slaves 133

6.4. Profinet 140
6.4.1. Profinet Client 140
6.4.1.2. Configuration Files Installation 140
6.4.1.3. Scan for Slaves 141
6.5. EtherCAT 147
6.5.1. EtherCAT Client 147
6.5.1.2. Configuration Files Installation 147
6.5.1.3. Scan for Slaves 148
7. Examples 155
7.1. Visualization 155
7.1.1. Create a new Visualization 155
7.1.2. Visualize the Scrolling LED 157
7.2. Remnant Variables 159
7.2.1. Retain Variables 159
7.2.2. Persistent Variables 160
7.2.3. Variable Behavior 161
7.3. Modbus TCP Client 163
7.4. Modbus TCP Server 167
8. Diagnosis and Troubleshooting 172
8.1. Error Notification 172
8.2. Log Information 172
8.3. Error ID 174

Chapter 1

1. Introduction

1.1. About This Manual

This document describes the use of the CODESYS programming environment and the RTE runtime system for the Advantech X86 series products.

Advantech provides add-on package for CODESYS which allows developers and end users to connected I/O modules; perform configurations, and simple testing of the I/O.

This manual supplies information about how to apply CODESYS to control Advantech X86 platforms, including software installation, writing a new program in CODESYS and how to use the fieldbus.

1.2. Organization of This Manual

This user manual is divided into the following sections:

- Introduction
- Installations
- Create and run a project
- Advantech I/O Modules
- Diagnosis and Troubleshooting

Introduction

This section gives the user a basic idea of this manual.

Installations

This section provides instructions on how to install CODESYS and Advantech Add-on Package

Create and run a project

This section gives the new user a walk-through in creating a simple program.

Advantech I/O Modules

This section introduces the detail configuration and mapping variables of Advantech APAX-5000 I/O modules and AMAX-5000 EtherCAT I/O Modules

Diagnosis and Troubleshooting

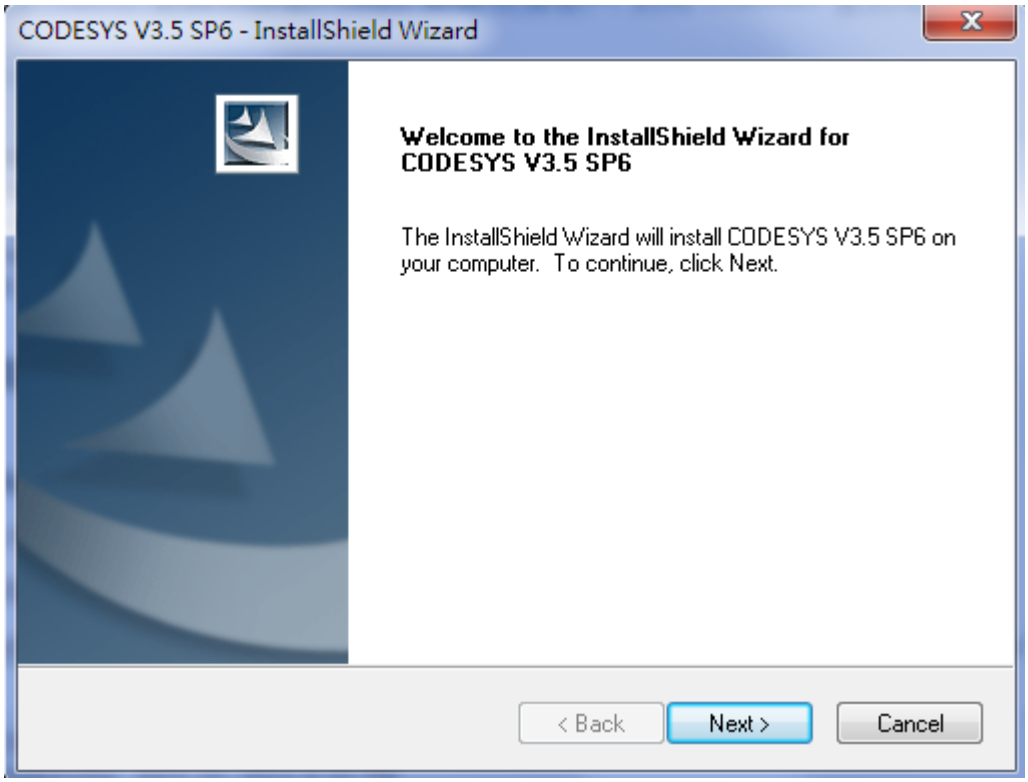
This section provides instructions on how to troubleshooting and diagnose operation mistakes or module errors.

Chapter 2

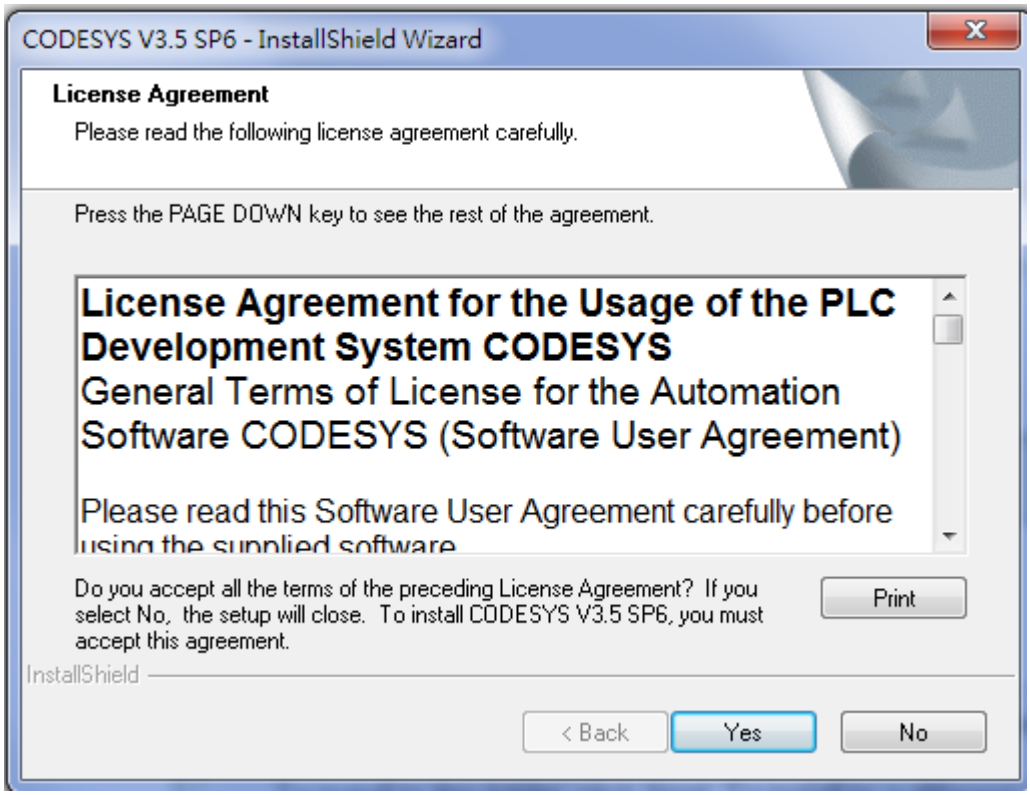
2. Installations

2.1. CODESYS Installation

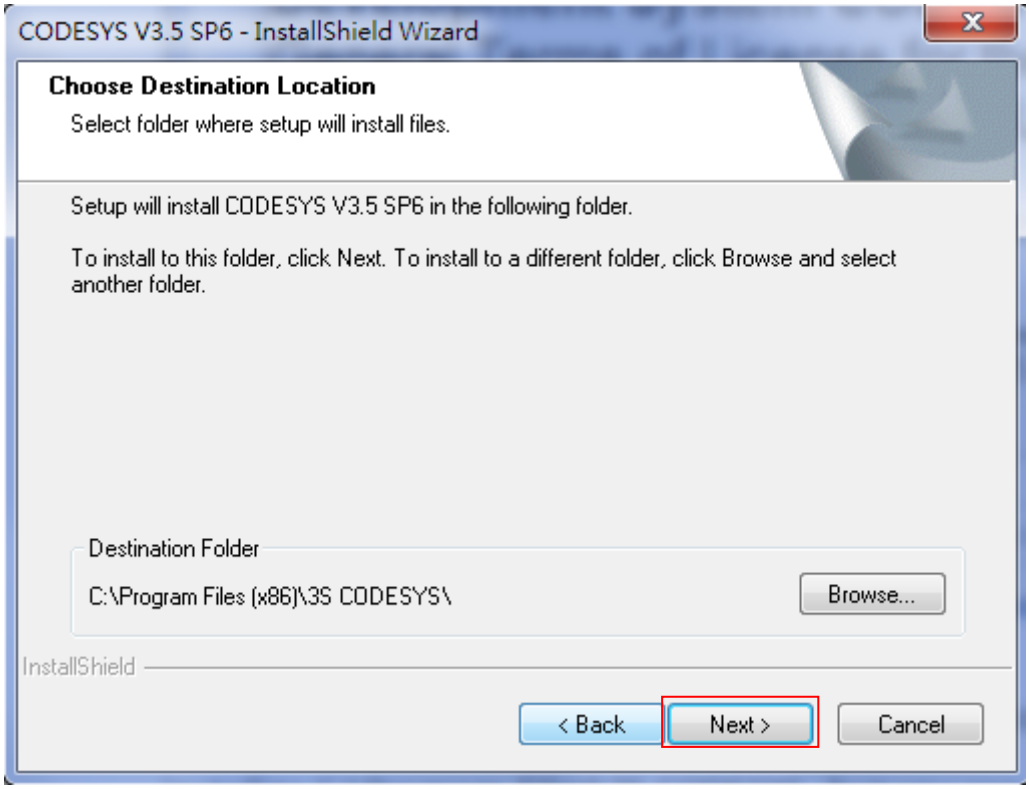
Step1: Double click and execute the "**Setup_CODESYSV<Version>.exe"** to start the installation assistant and then click Next to continue.



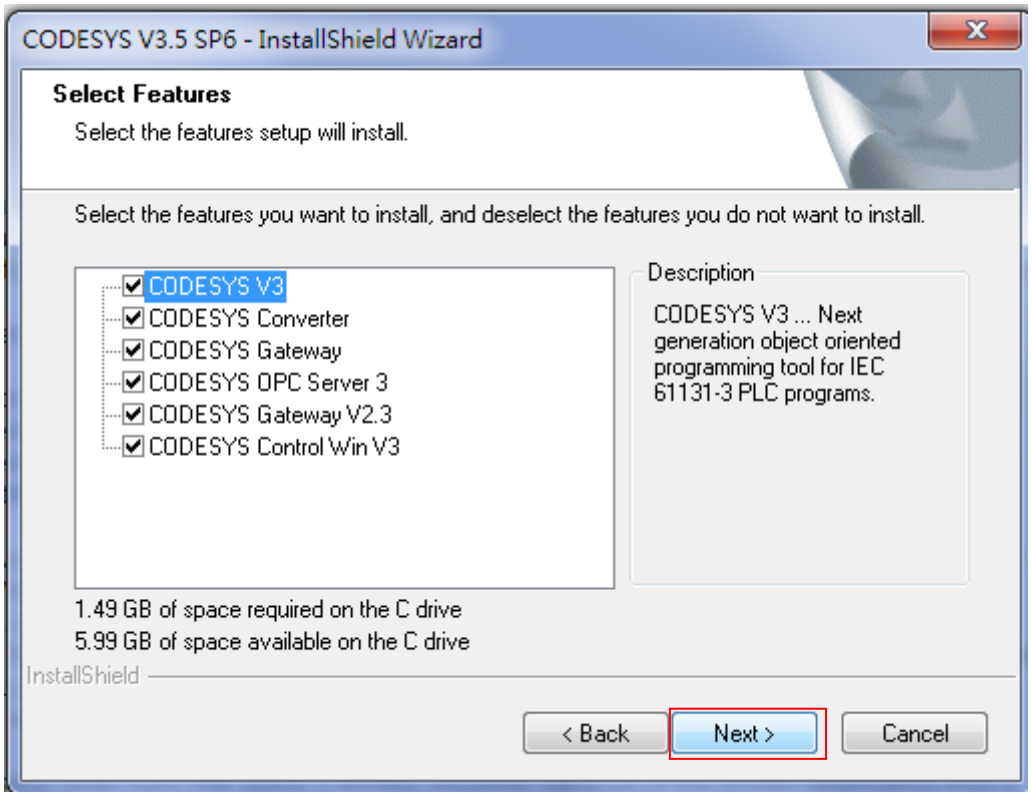
Step2: You must accept License Agreement



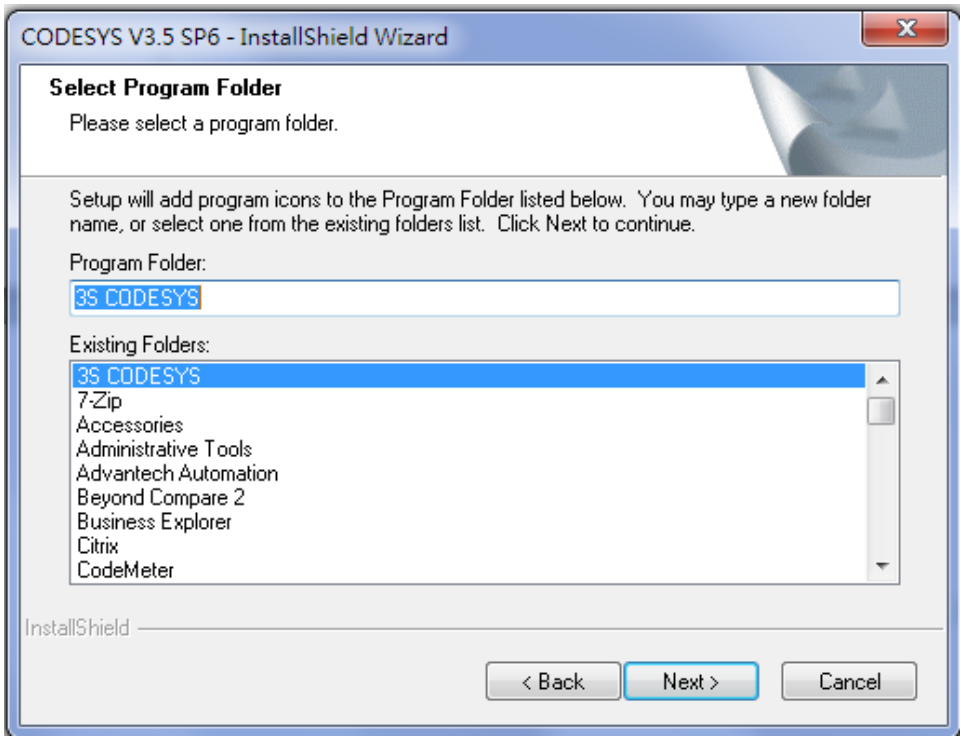
Step 3: You will then be prompted for the installation location. By default, CODESYS will install to C:\Program Files\3S CODESYS, but you can specify the location or folder name of your choice. Click Next to proceed.



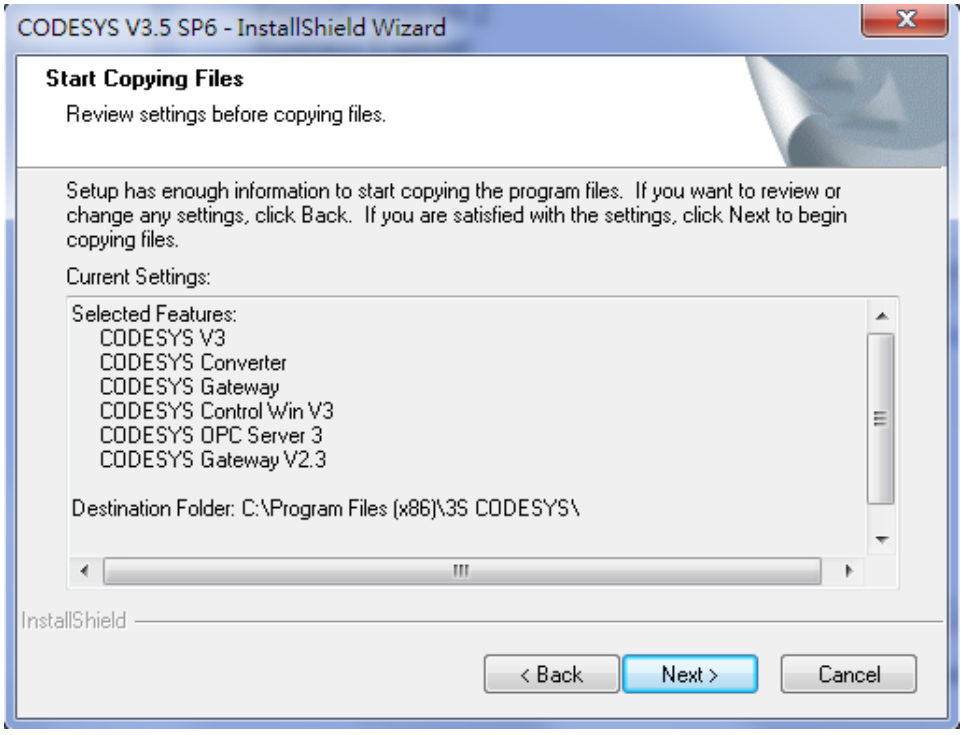
Step 3: Select all features and then click Next to proceed.



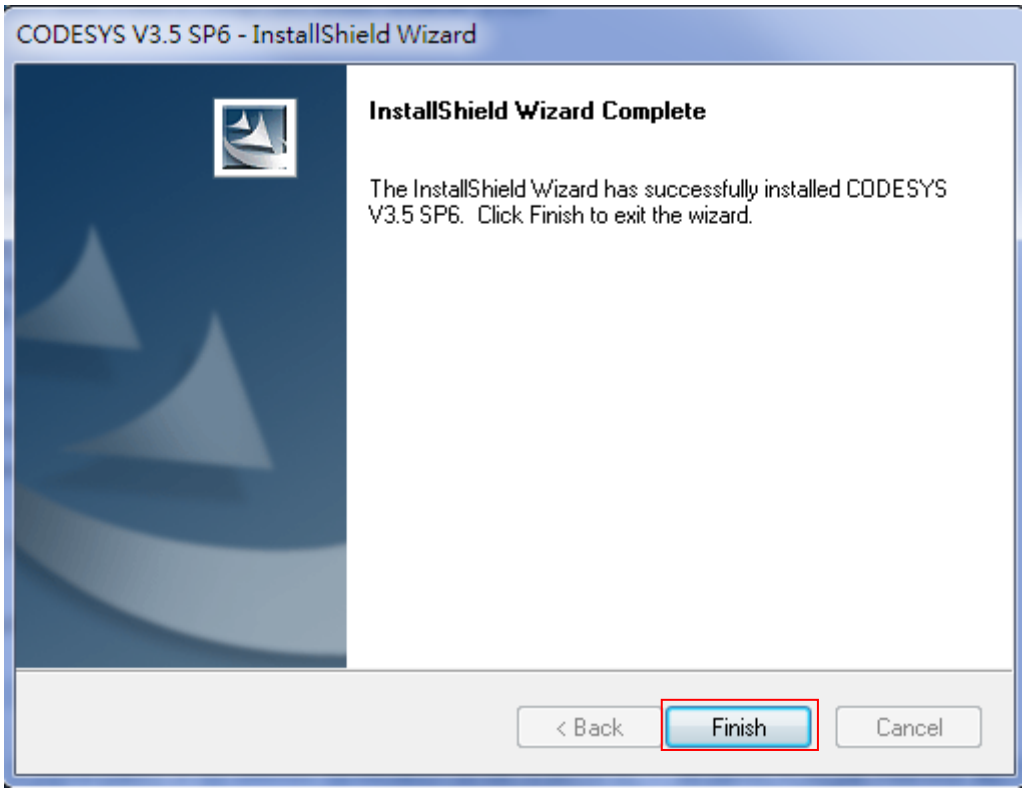
Step4: You must the program folder “3S CODESYS”. Please do not change and then click “Next” to proceed.



Step5: Start coping files and then click “Next” to proceed.



Step 6: Complete to install CODESYS and you'll see CODESYS icon  which is available on the desktop. Click "Finish" to close the installation wizard.



2.2. Add-on Package Installation

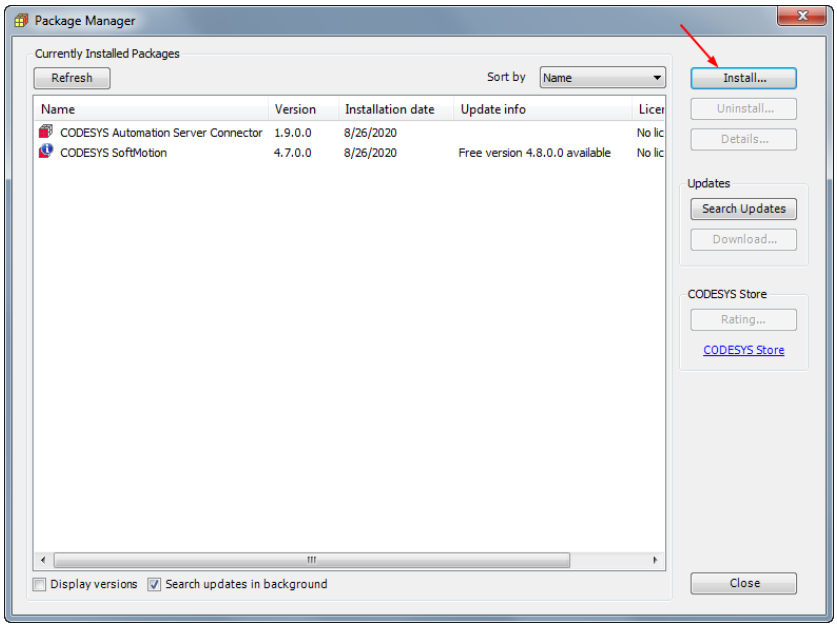
Now you have installed CODESYS on your system successfully. To equip Advantech device information or use Advantech add-on functionalities, user needs to follow below steps to install Advantech add-on Package on the CODESYS.

2.2.1. Installation

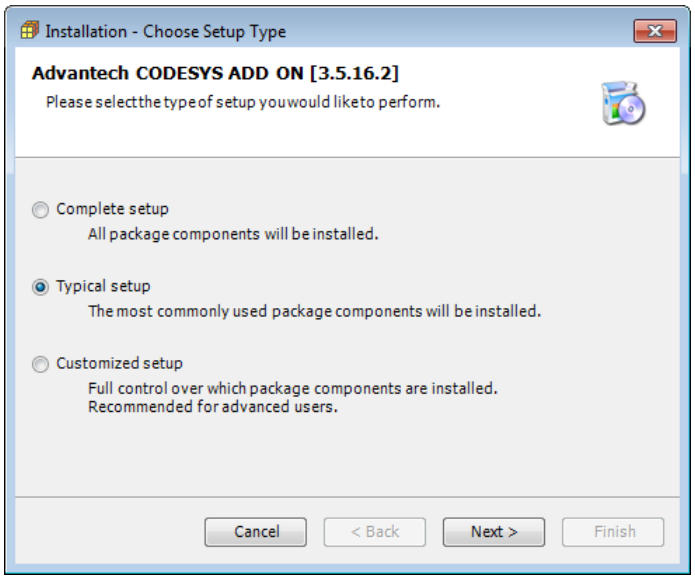
Step 1: Install add-on package from CODESYS Package Manager. Activate the dialog window from the menu (**Tools -> Package Manager**)

Step 2: In the Package Manager, click Install and select the file. The file would be as below format.

"**Advantech CODESYS ADD ON V<Version>.package"**.



Step 3: Select Typical setup and click next.



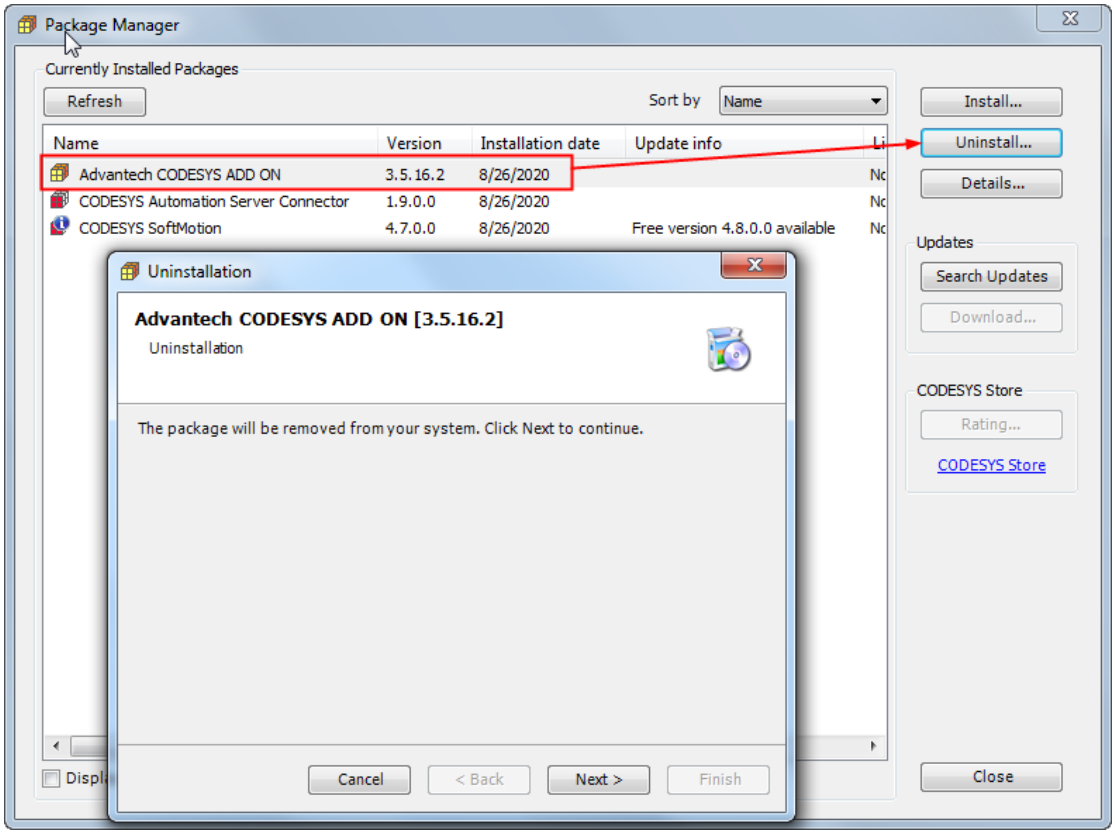
Step 4: After few dialog window, you will see the completion screen. Click "Finish" to close the installation wizard.

2.2.2. Updating the Package

It's highly recommended that you uninstall the previous version package before updating and installing new add-on package.

Start CODESYS and perform command **Package Manager** from the menu (**Tools -> Package Manager**). Select the package you want to uninstall and then click “Uninstall”. Click "Close" to close the package manager.

After uninstalling the old package successfully, please refer to Chapter 2.2.1 Installation



Chapter 3

3. Create and run a project

3.1. Start CODESYS

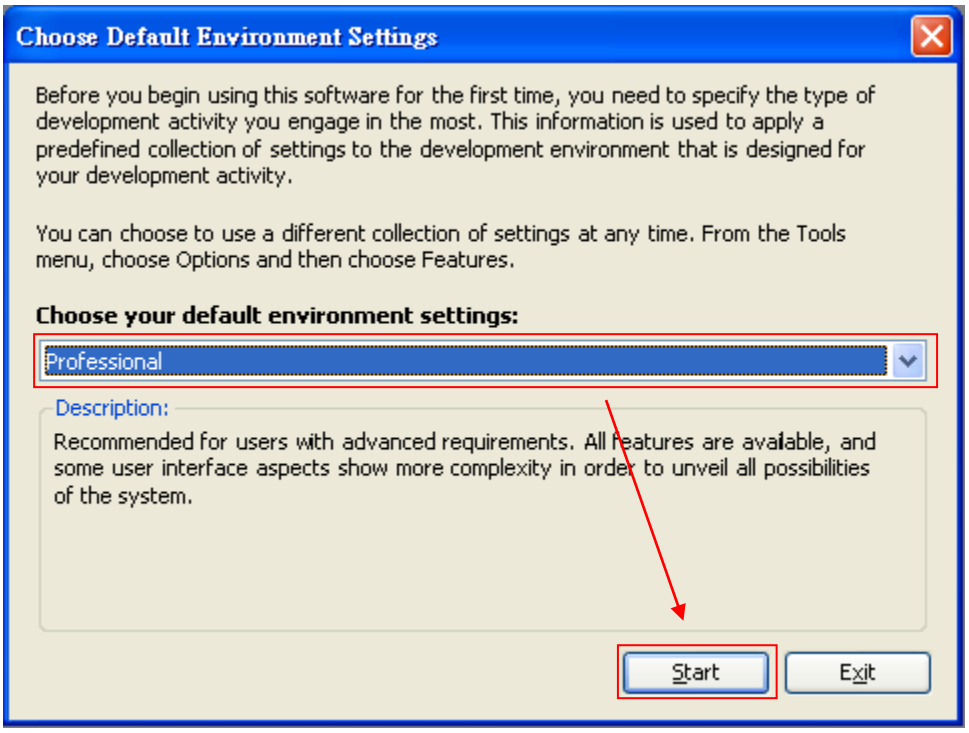
Start CODESYS by double-clicking the CODESYS icon 

which is available on the desktop.

Alternatively, you can start the CODESYS programming system with

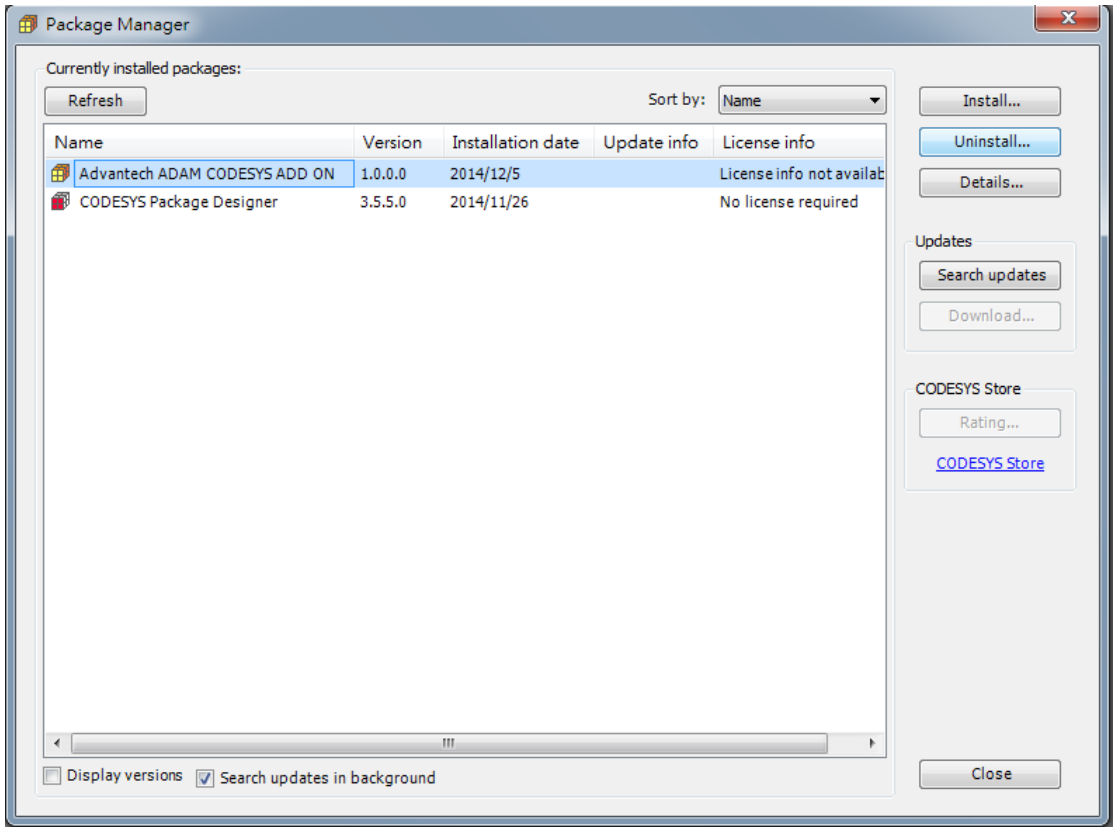
Start -> Programs -> 3S Software -> CODESYS -> CODESYS V<version>

When you start the programming system the first time after first installation on the system, you will be asked to choose the default collection of settings and features. Choose the "**Professional**" and then click Start to proceed.



Before creating a project, make sure that Advantech ADAM add-on package is installed successfully. Choose **Package Manger** from the **Tools** menu:

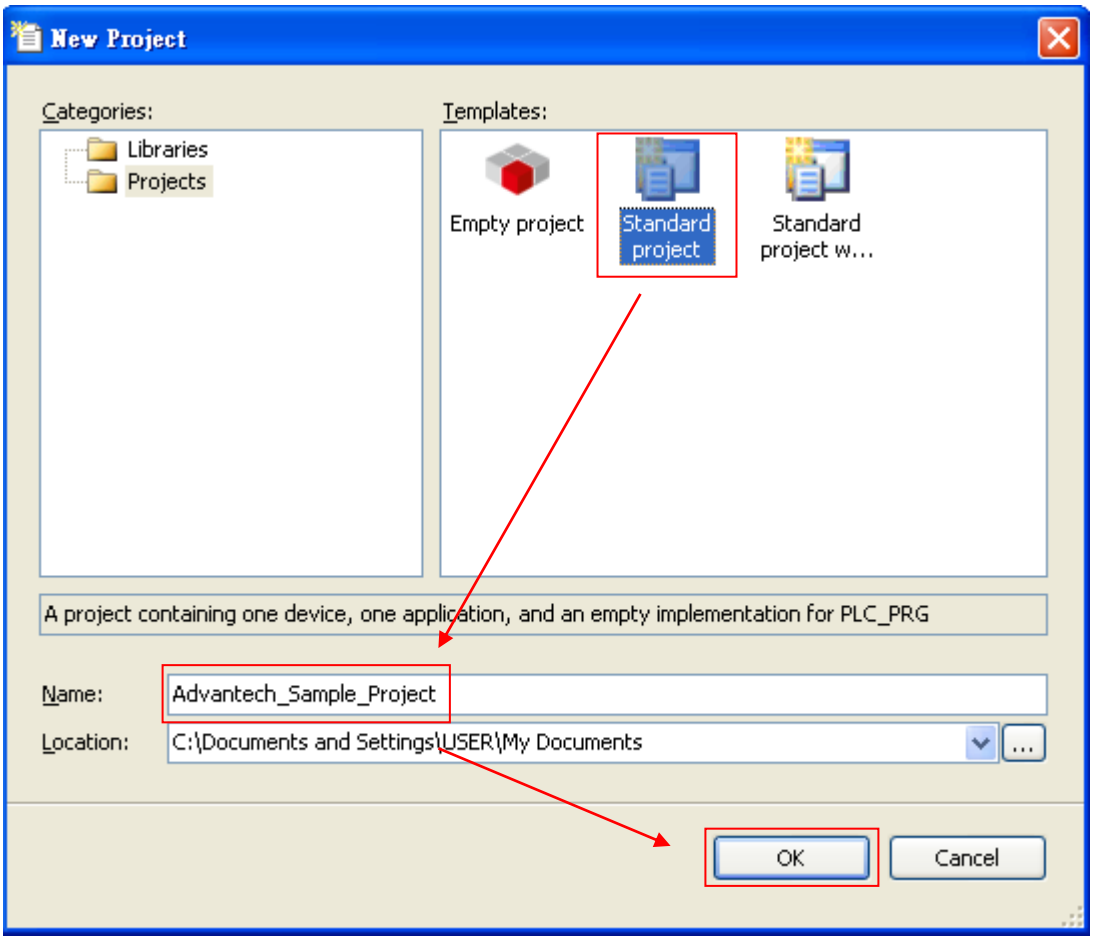
If the Advantech ADAM add-on package didn't show in manager, please refer to Chapter 2 and update your package.



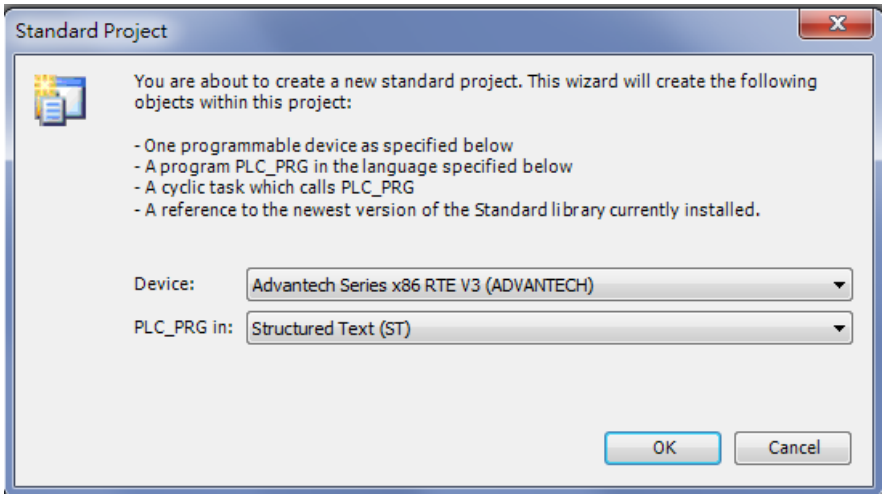
3.2. Create a Project

Step 1: To create a new project, choose command **New project** from the **File** menu:

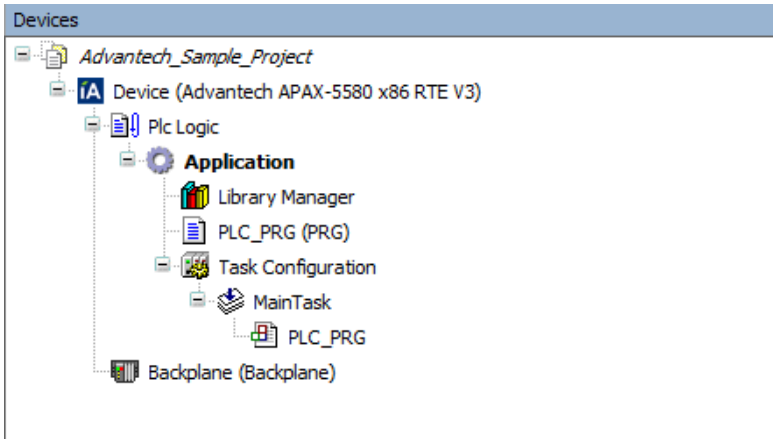
In the New Project dialog select **Standard project** in the 'Templates' field and enter a Name and a Location path for the project file. Press OK to confirm.



Step 2: You will then be prompted for choosing devices. Choose device **Advantech Control x86 RTE V3 (ADVANTECH)** and programming language **Structured Text (ST)** (depend on developer) for PLC_PRG. Press OK to open the new project.

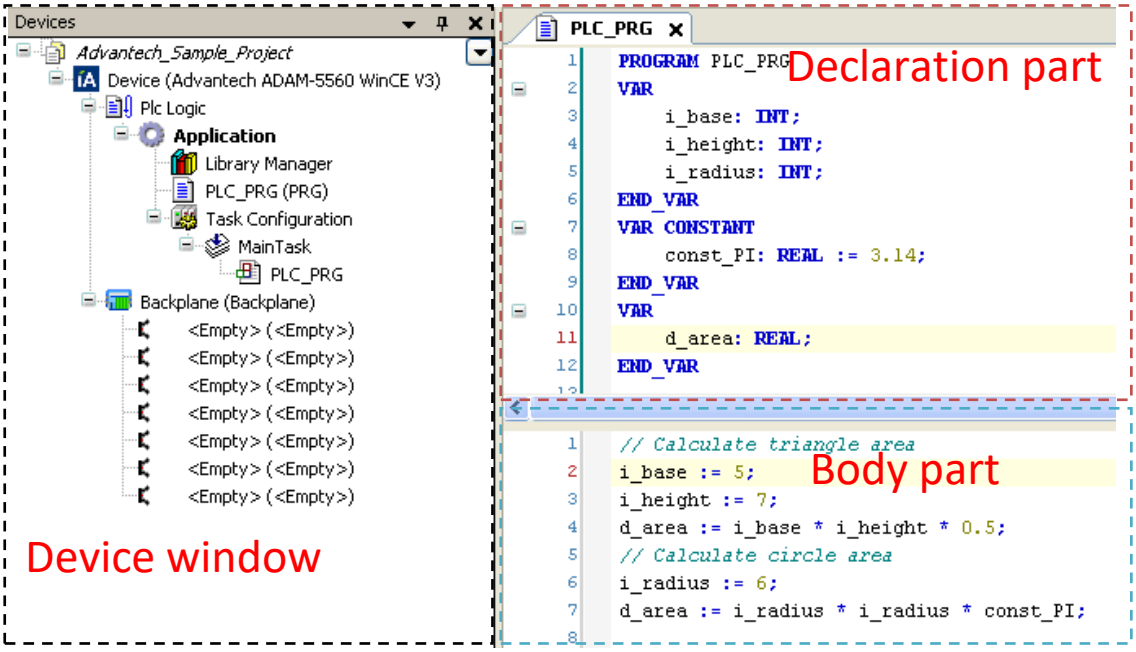


Step 3: The project name now will appear in the title bar of the CODESYS user interface and the Devices window.



3.3. Write a Program

In the **Devices** window, double-chick **PLC_PRG(PRG)** and language editor window will open. The editor consists of a declaration part (upper) and a body part (lower), separated by a screen divider. The declaration part shows line numbers at the left border and the embracing keywords "VAR" and "END_VAR" for the variables declaration.



In the declaration part of the editor put the cursor behind VAR and press the Return-key. A new empty line will be displayed where you enter the declaration of variables.

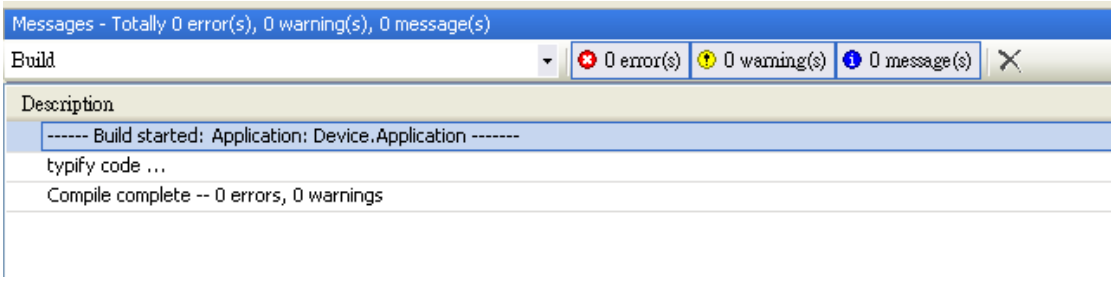
Here, we write a simple program to calculate the area of triangle and circle, so declare **i_base, i_height, i_radius** which are of type INTEGER, **d_area** of type REAL:

```
PROGRAM PLC_PRG
VAR
    i_base: INT;
    i_height: INT;
    i_radius: INT;
END_VAR
VAR CONSTANT
    const_PI: REAL := 3.14;
END_VAR
VAR
    d_area: REAL;
END_VAR
```

In the body part of the PLC_PRG editor put the cursor in line 1 and enter the following lines:

```
// Calculate triangle area
i_base := 5;
i_height := 7;
d_area := i_base * i_height * 0.5;
// Calculate circle area
i_radius := 6;
d_area := i_radius * i_radius * const_PI;
```

We need to check the program for syntactic errors and perform command **Build**  from the menu (**Build -> Build**) or press **<F11>**:



Note!

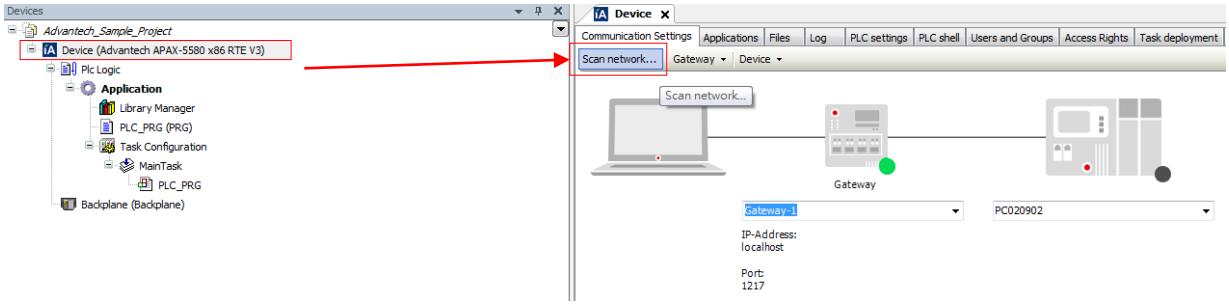
No code will be generated in this case. Error messages will be displayed in the Messages window which is placed at the lower part of the user interface per default.

3.4. Connect to the Target Device

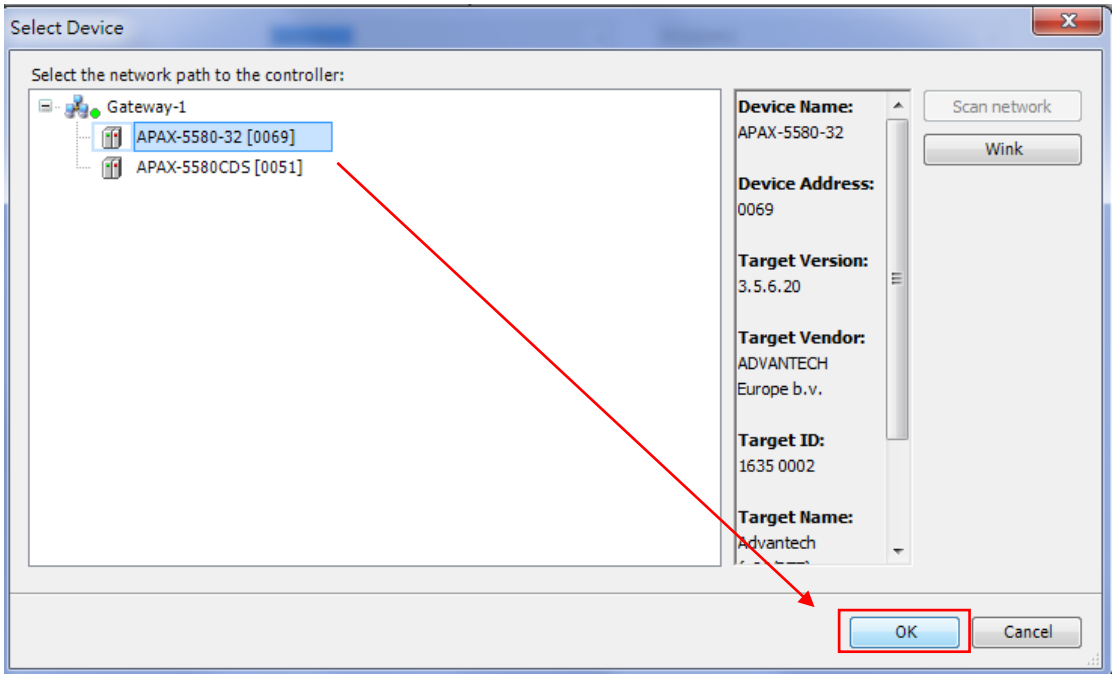
In this section, we want to discuss how to connect to Advantech X86 RTE platforms.

We need to set the active application by using Device editor. It displays an icon of programming device, the current gateway and the target device with their connection status.

The **Device editor** opens by double clicking the device name in the device tree.



Click the **Scan network** button to search for available devices in your local network. You will then be prompted for the device selection. Choose your target device and click OK to proceed.



In Device editor, it will show the connection status. Please check that the colored status points are all in green.

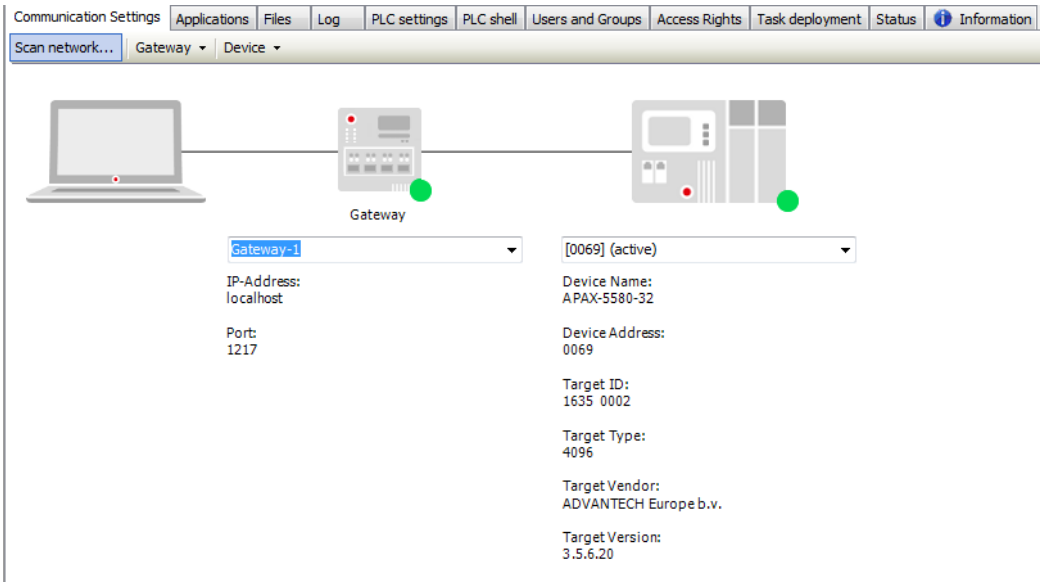
Note!

Meaning of the colored status point on the gateway and the device:

Red: Connection cannot be established

Green: Connection established

Black: Connection not defined



3.5. Run the Application

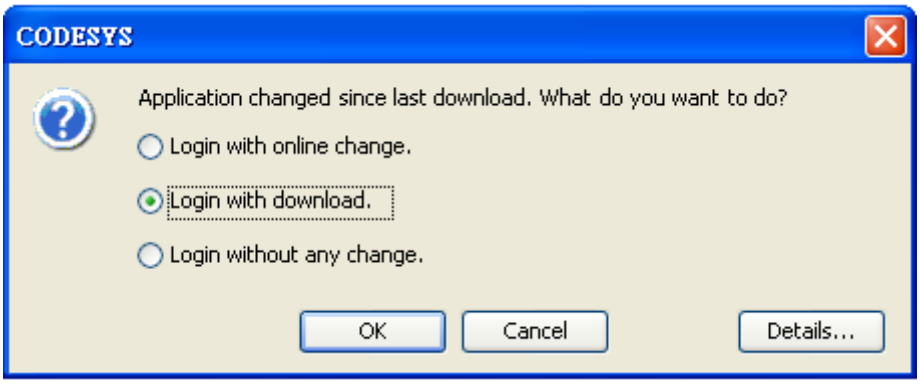
We can download the application by performing command **Login**  from the menu (**Online -> Login**) or press **<Alt+F8>**. You will then be prompted for choosing login options. Here, we choose “Login with download” for the first time and click OK to proceed.

Note!

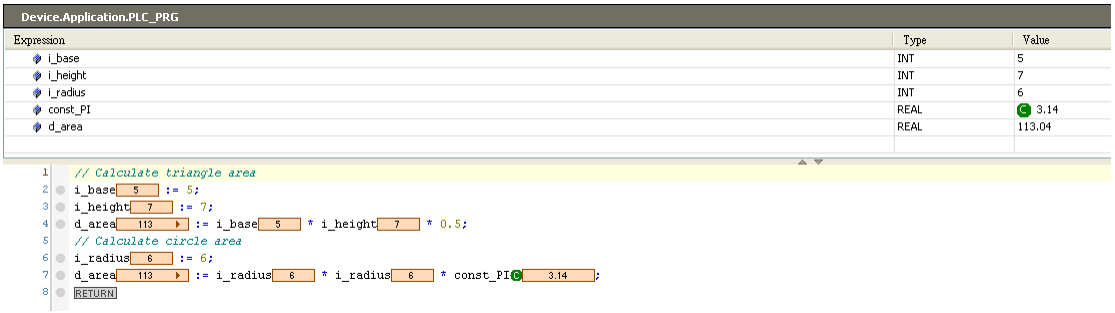
Meaning of login options:

“Login with online change”: Only the modified objects will be loaded.

“Login with download”: The complete application will be loaded and initialized completely. “Login without any change”: The latest modifications will not be loaded.



We run the program by performing command **Start** from the menu **(Debug -> Start)** or press **<F5>**. The online view of PLC_PRG will be opened: In the upper part a table shows the watch variables in application. In the lower part you see the code lines as entered in offline mode, supplemented by the little inline monitoring windows behind each variable, showing the actual value.



Stop the program by performing command **Stop** from the menu (**Debug -> Stop**) or press **<Shift+F8>**.

If you want to change into the offline mode and disconnect the programming system from the target device, perform command **Logout** from the menu (**Online -> Logout**) or press **<Ctrl+F8>**.

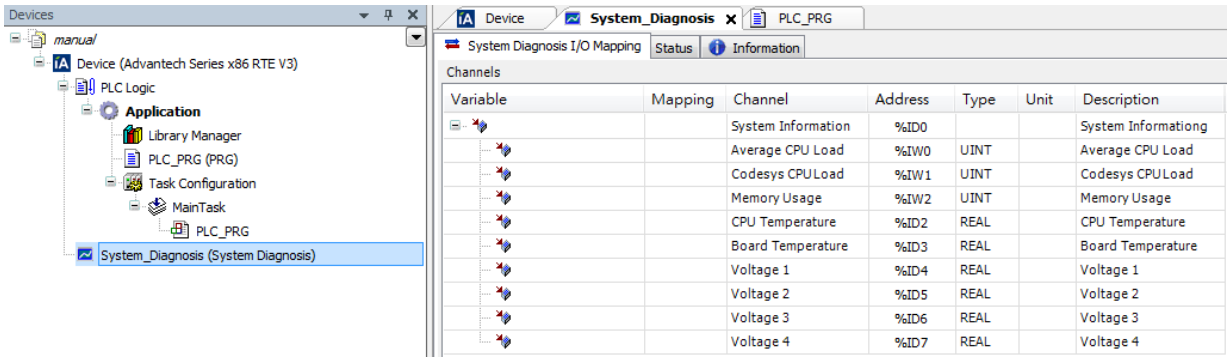
Chapter 4

4. System Diagnosis

4.1. System Information

For diagnosis, we provide the related system information including CPU load, memory usage, CPU temperature, etc.

Click the **System Diagnosis,** the detail of the system information is shown.



Average CPU Load: the average percentage of CPU load. For example, if there are dual core, it is the average percentage of dual core's load.

CODESYS CPU Load: the percentage of the load of CPU CODESYS is running without Windows existed.

Memory Usage: the percentage of memory is in active use.

CPU Temperature: the temperature of the processor.

Board Temperature: the temperature of the main board.

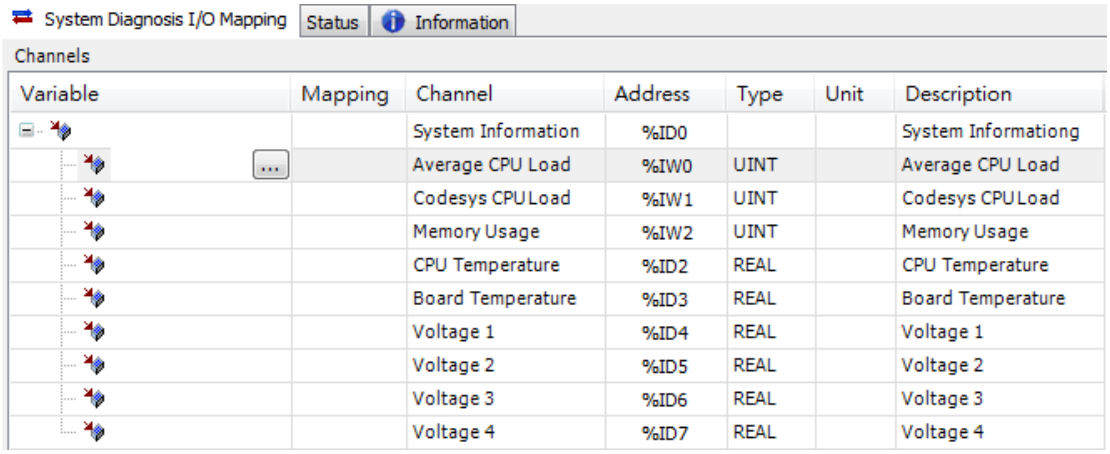
4.2. Map Variables to System Information

In this section, we want to discuss how to map variable to system information for programming use. For more details on creating a new program please refer to Chapter 3. Here, we declare *uiLoad* and *bAlarm* in declaration part and *bAlarm* is set to true if *uiLoad* is more than 50 in body part.

```
PROGRAM PLC_PRG
VAR
    uiLoad: UINT;
    bAlarm: BOOL := FALSE;
END_VAR
```

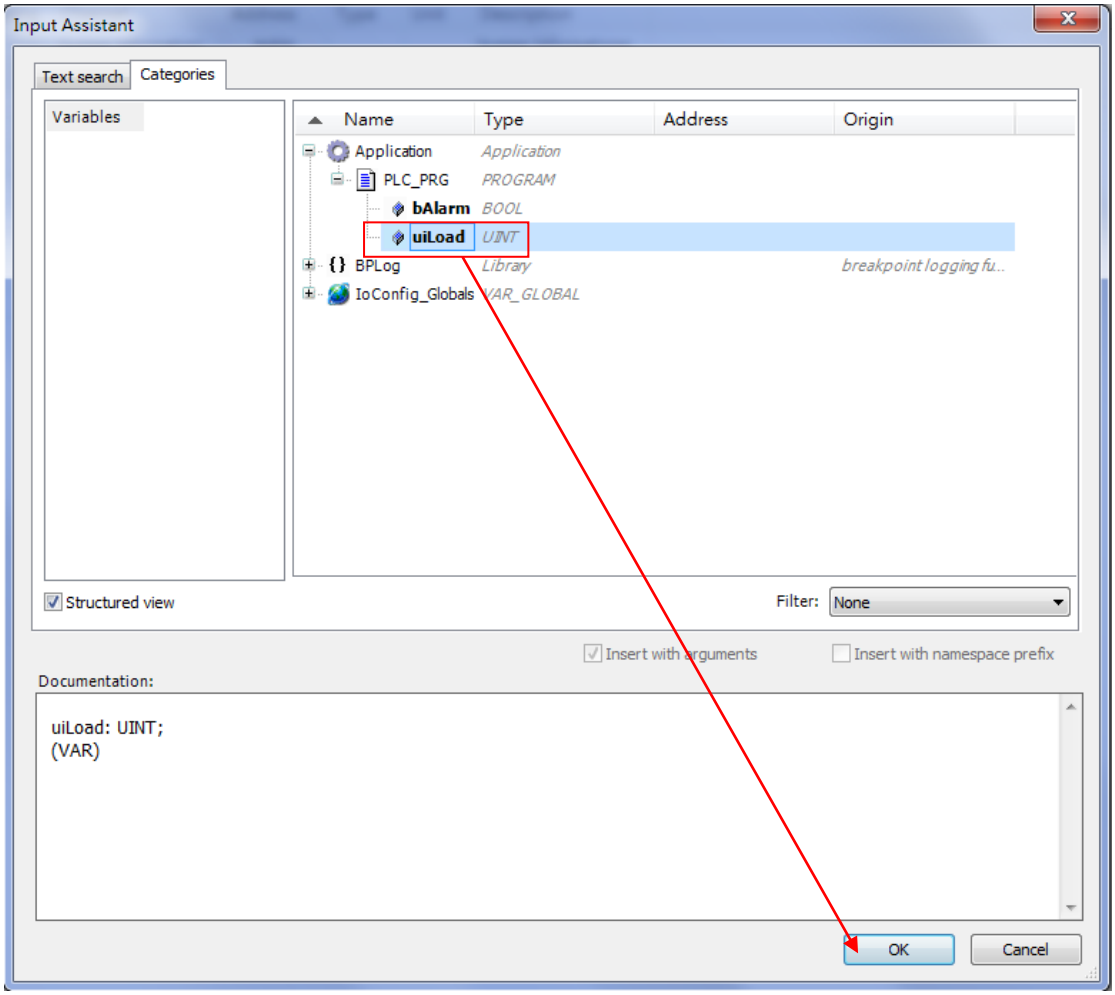
```
IF uiLoad > 50 THEN
    bAlarm := TRUE;
END_IF
```

Open **System Diagnosis** by double clicking in the device tree. Double-click on the variable column and choose mapping variable by clicking the button [...]. In this example, we try to map the variable (*uiLoad*) to the average CPU load, so we double-click on the first row of variable column.



Variable	Mapping	Channel	Address	Type	Unit	Description
		System Information	%ID0			System Informationg
...		Average CPU Load	%IW0	UINT		Average CPU Load
		Codesys CPU Load	%IW1	UINT		Codesys CPU Load
		Memory Usage	%IW2	UINT		Memory Usage
		CPU Temperature	%ID2	REAL		CPU Temperature
		Board Temperature	%ID3	REAL		Board Temperature
		Voltage 1	%ID4	REAL		Voltage 1
		Voltage 2	%ID5	REAL		Voltage 2
		Voltage 3	%ID6	REAL		Voltage 3
		Voltage 4	%ID7	REAL		Voltage 4

It will open the **Input Assistant Dialog**, where you can choose one of available variables stored for the average CPU load.



Now, we can download the application by performing command **Login** and then performing command **Start**.

```
IF uiLoad 11 > 50 THEN
    bAlarm FALSE := TRUE;
END_IF RETURN
```

Chapter 5

5. Advantech I/O Modules

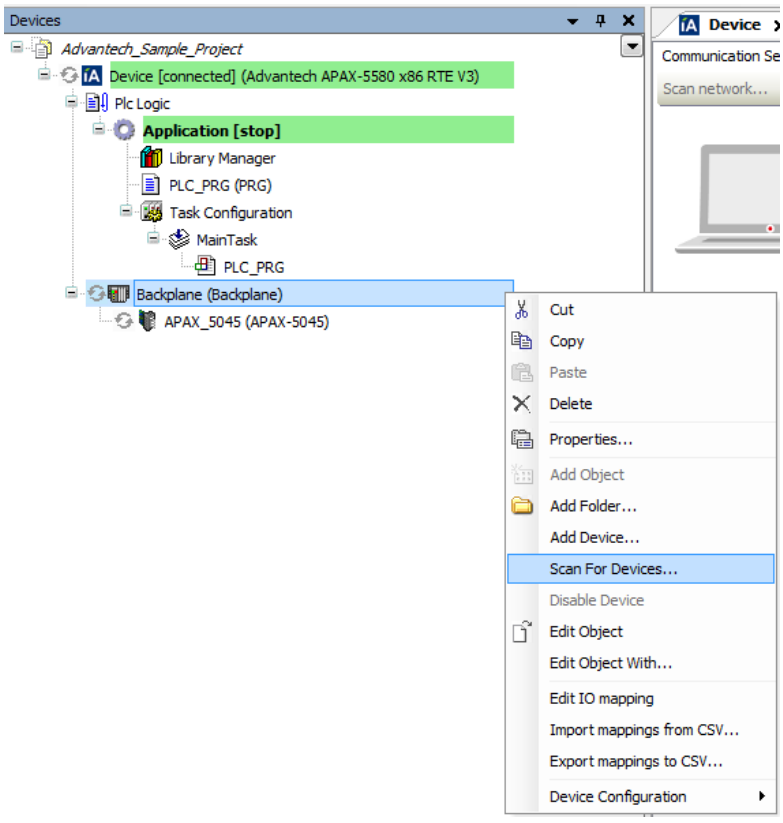
5.1. APAX-5000 IO Modules

5.1.1. Scan I/O Modules into CODESYS

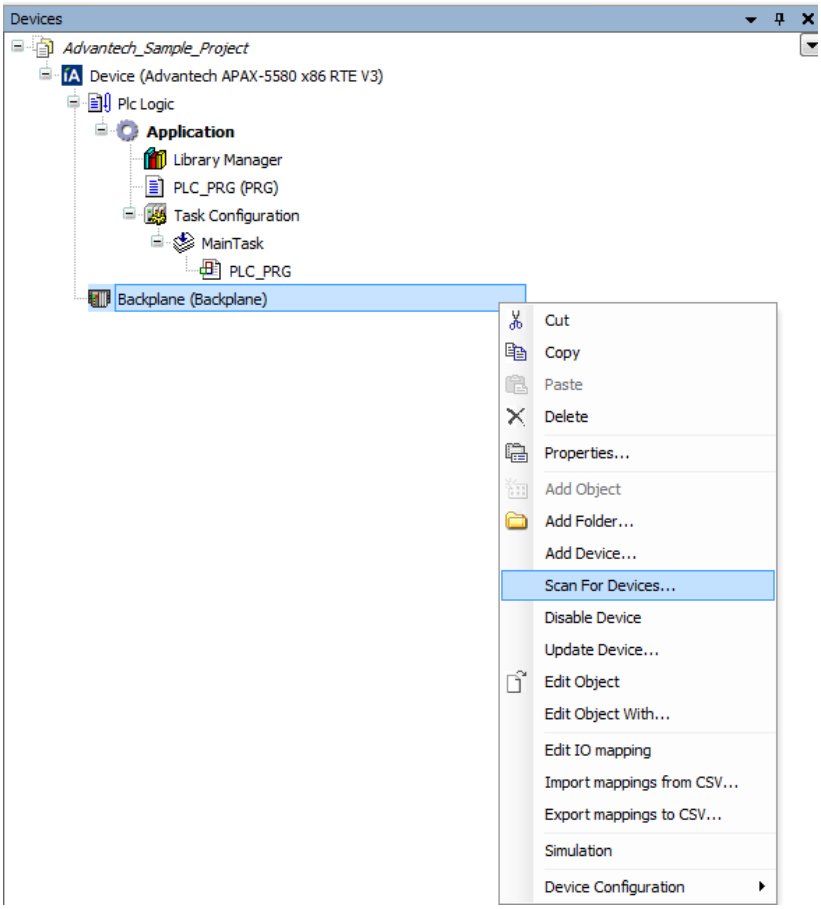
We can scan and configure Advantech APAX-5000 I/O modules as objects in the device tree.

If there is no any project existed in APAX-5580, you must login first.

Choose the **Backplane** and click **Scan For Devices** in context menu.



If there is a project existed in the APAX-5580, you can directly click **Scan For Devices** in context menu.



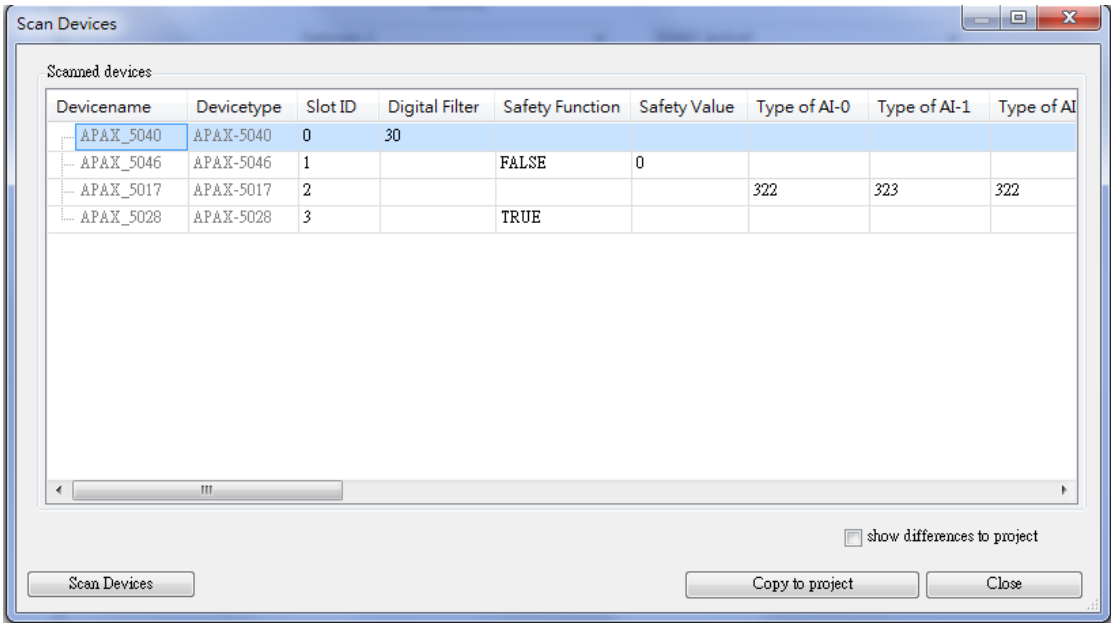
And then you can get all the online APAX-5000 IO modules. Copy the specified IO module or all IO modules to the project by click **Copy to project** or **Copy all devices to project**.

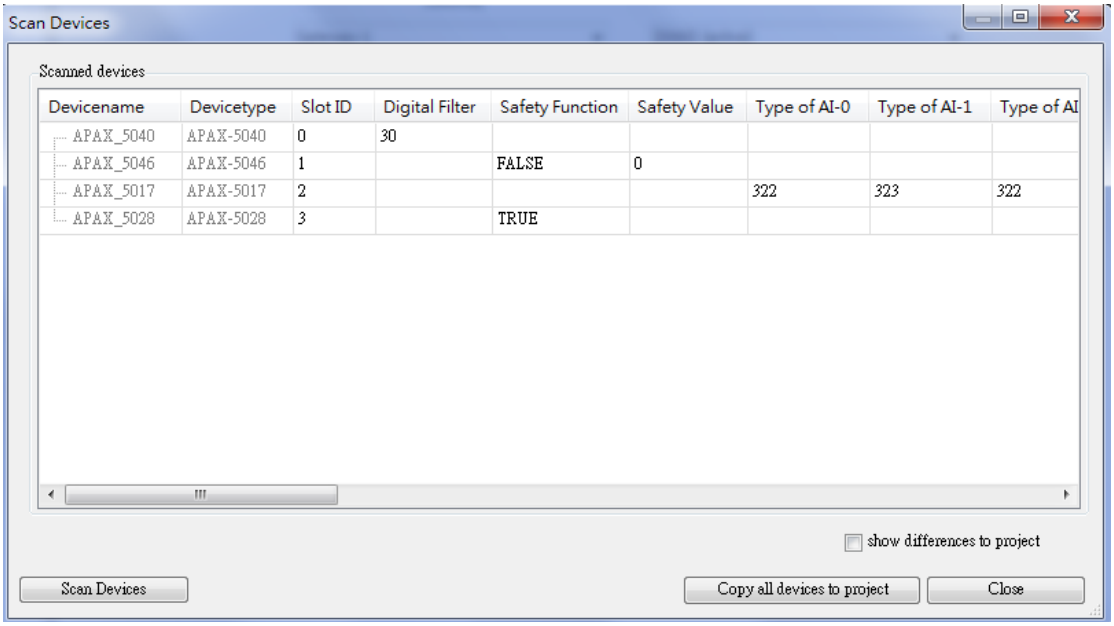
Note!

(1) Make sure that the APAX-5580 is not in RUN mode before scanning IO.

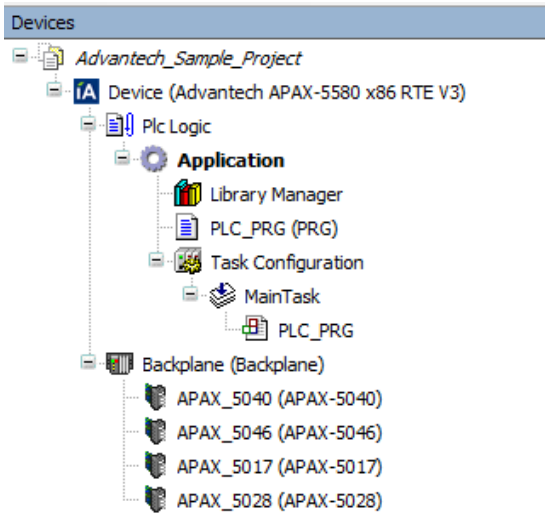
Only when APAX-5580 is in STOP mode, scan IO is available.

(2) Through scanning IO modules, there is a limitation that cannot get current safety value of each channel from APAX-5028. Therefore, please re-configure the **safety value** of each channel of APAX-5028 after scanning IO modules.





All the devices or the specified device will be added into the Backplane.



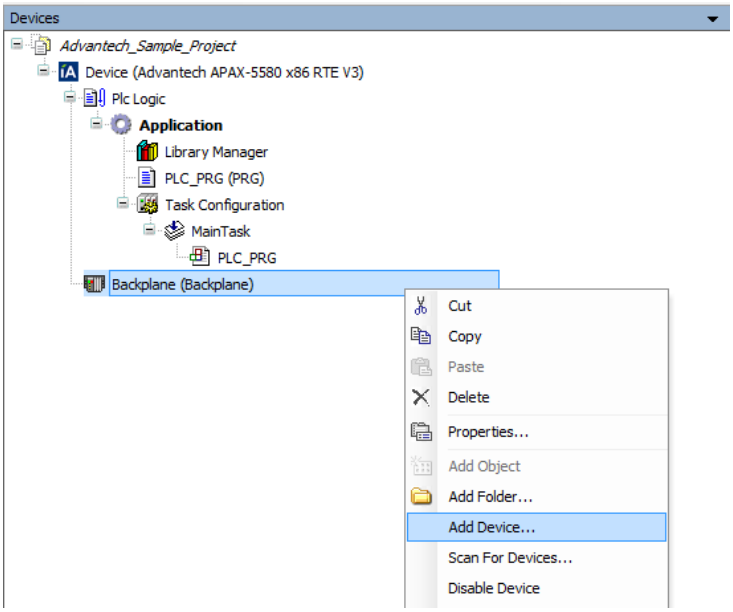
Note!

You can remove the existing device by click **Delete** in context menu.

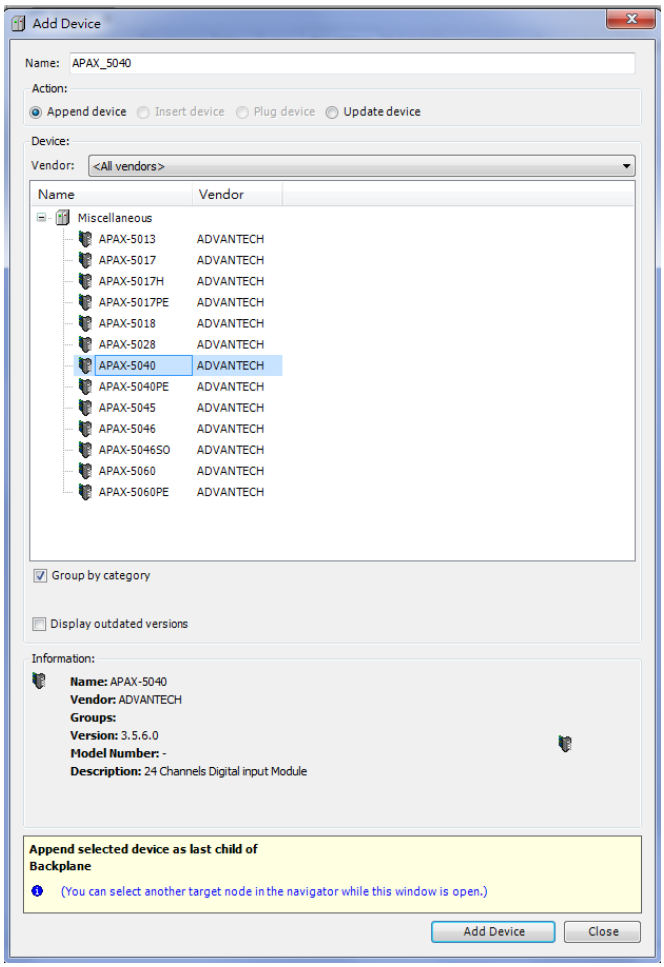
5.1.2. Insert I/O Modules into CODESYS

We can add and configure Advantech APAX-5000 I/O modules as objects in the device tree.

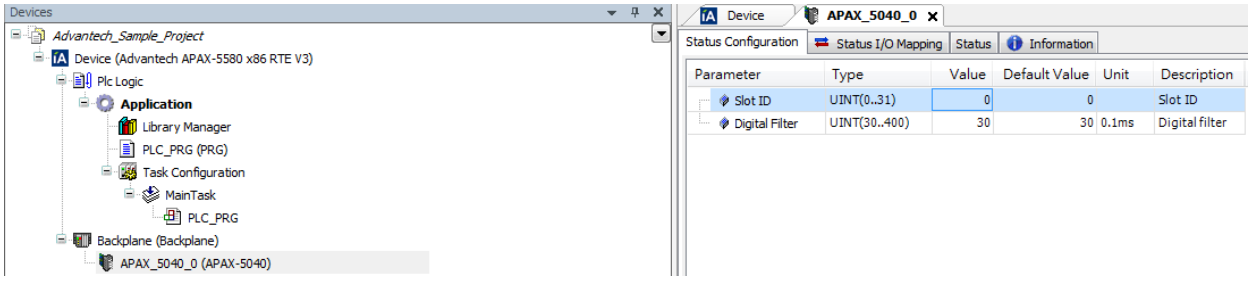
Choose the **Backplane** and click **Add Device** in context menu.



It will open the **Add Device dialog**, where you can choose one of available devices. Click **Add Device** to proceed and then press **Close** to close the device dialog.



Then the specified IO module will be added into the Backplane.



Note!

If inserting IO modules is used, you must configure the right Slot ID of each inserted APAX-5000 IO modules. If scanning IO modules is used, you do not need to do the Slot ID configuration.

5.1.3. Map Variables to I/O Modules

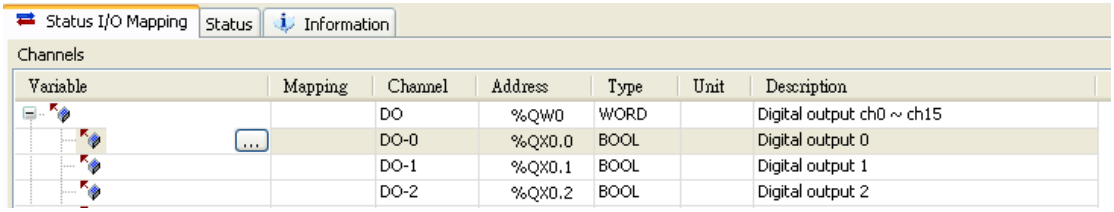
In this section, we want to discuss how to map variable of program to Advantech I/O modules. For more details on creating a new program please refer to Chapter 3.

Here, we declare *bValue* in declaration part and set true in body part.

```
PROGRAM PLC_PRG
VAR
    bValue: BOOL;
END_VAR
```

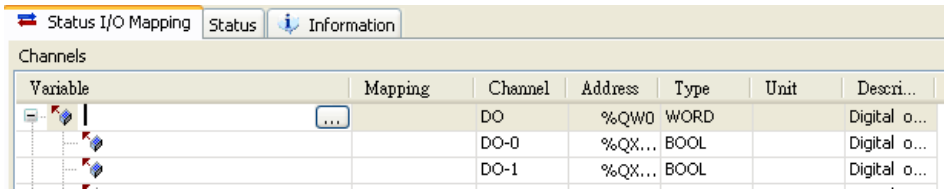
```
bValue:= true;
```

Open **Module Editor** by double clicking the device name in the device tree. Double-click on the variable column and choose mapping variable by clicking the button [...]. In this example, we try to map the variable (*bValue*) to channel 0, so we double-click on the first row of variable column.

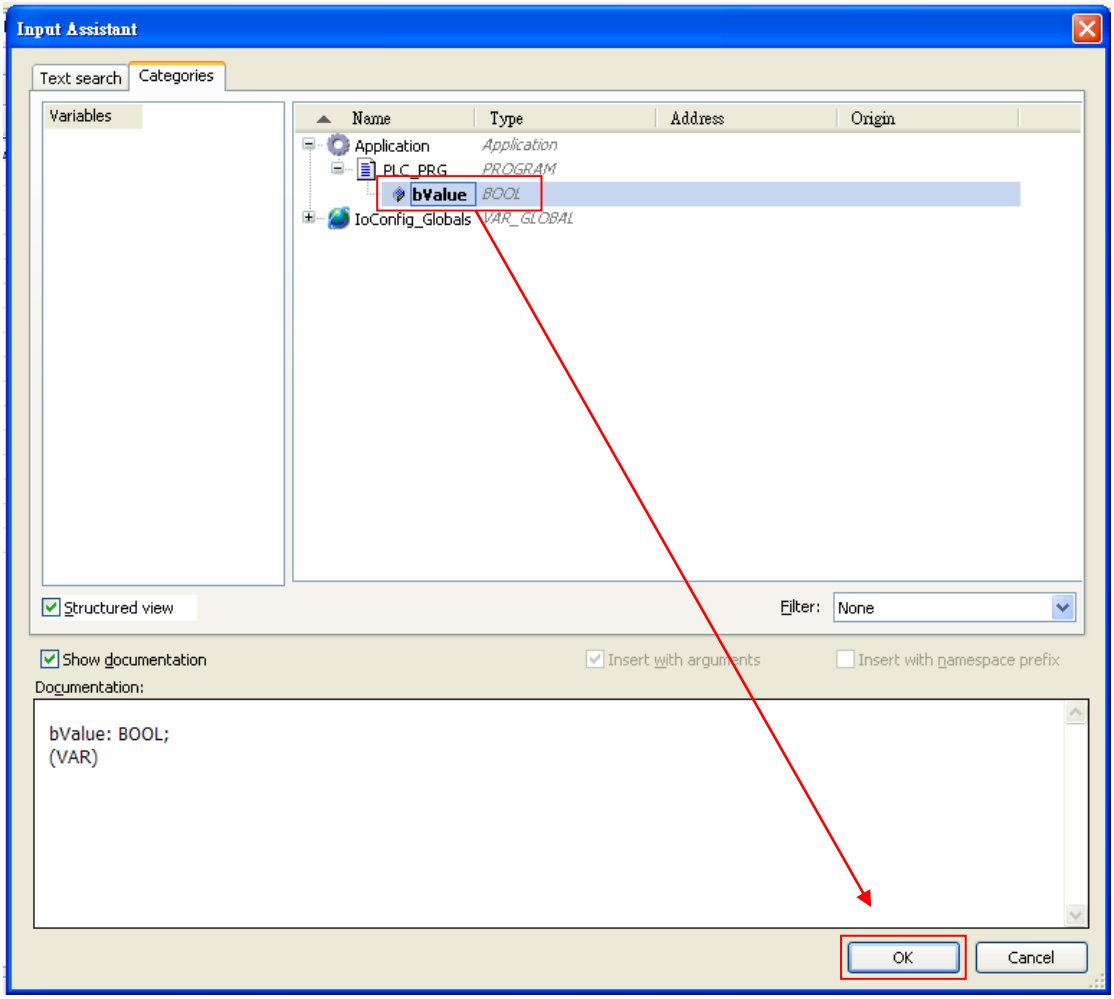


Note!

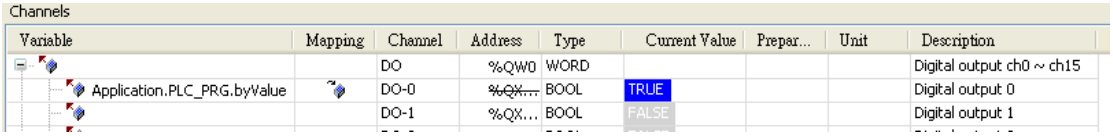
If you want to control all channels, declare a WORD variable and map it.



It will open the **Input Assistant Dialog**, where you can choose one of available variables for the current digital output channel.

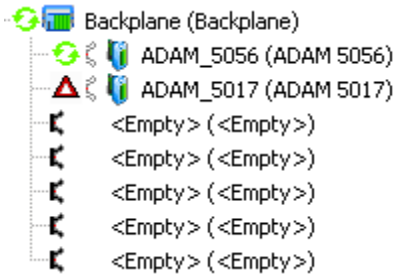


Now, we can download the application by performing command **Login** and then performing command **Start**. The channel-0 of I/O module will be lit up.



Note!

If the Advantech modules are correctly configured, it will show a green circle icon next to the device name in the device tree. If it shows a red triangle , see Chapter 8 for troubleshooting.



5.1.4. Support List

Advantech provides 13 types of APAX-5000 I/O modules for various applications so far. Following table is the I/O modules support list. In the following section, we will introduce I/O modules according to their types.

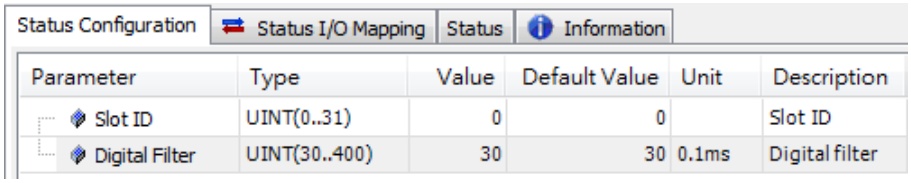
Module	Name	Specification	Reference
Analog Input	APAX-5013	8-ch RTD Input	Isolated
	APAX-5017	12-ch AI	Isolated
	APAX-5017H	12-ch High Speed AI	Isolated
	APAX-5017PE	12-ch AI	Isolated
	APAX-5018	12-ch TC Input	Isolated
Analog Output	APAX-5028	8-ch AO	Isolated
Digital Input	APAX-5040	24-ch DI w/LED	Isolated
	APAX-5040PE	24-ch DI w/LED	Isolated
Digital Output	APAX-5046	24-ch DO w/LED	Isolated
	APAX-5046SO	20-ch DO source w/LED	Isolated
Digital I/O	APAX-5045	24-ch DI/O w/LED	Isolated
Relay Output	APAX-5060	12-ch Relay Output w/LED	Isolated
	APAX-5060PE	12-ch Relay Output w/LED	Isolated

5.1.5. Digital Input Modules

In this section, we are going to introduce digital input modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration, Status I/O Mapping, Status and Information.**

Status Configuration: Provide the status page for setting device. Double-click on the value column of the particular setting.



Note!

If scan IO and then copy to the project is used, do not need to modify the Slot ID.

If manually insert IO into the project is used, need to configure the right Slot ID.

Status I/O Mapping: Show the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in single bit. If the value is “true”, it means that the channel is on; “false” for off. All channel values can represent as one word.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 24 digital inputs. This will require either 24 Boolean addresses or 2 WORD (4 BYTE) addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

I = Physical Input

W = Word (16 bits)

X = Single bit

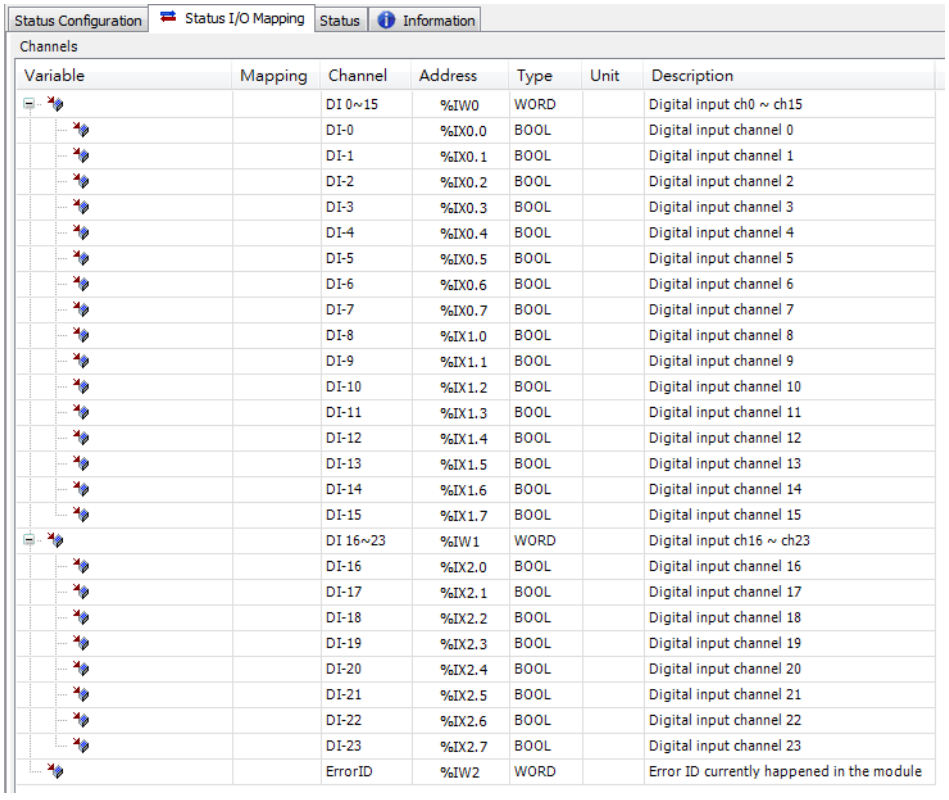
$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

Status: The reserved page.

Information: Provide the brief information to current module.



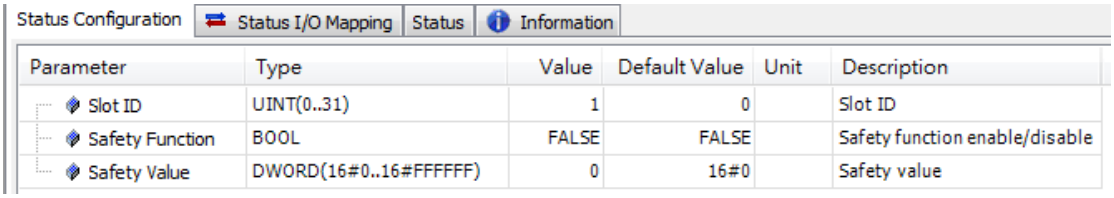
Variable	Mapping	Channel	Address	Type	Unit	Description
		DI 0~15	%IW0	WORD		Digital input ch0 ~ ch15
		DI-0	%IX0.0	BOOL		Digital input channel 0
		DI-1	%IX0.1	BOOL		Digital input channel 1
		DI-2	%IX0.2	BOOL		Digital input channel 2
		DI-3	%IX0.3	BOOL		Digital input channel 3
		DI-4	%IX0.4	BOOL		Digital input channel 4
		DI-5	%IX0.5	BOOL		Digital input channel 5
		DI-6	%IX0.6	BOOL		Digital input channel 6
		DI-7	%IX0.7	BOOL		Digital input channel 7
		DI-8	%IX1.0	BOOL		Digital input channel 8
		DI-9	%IX1.1	BOOL		Digital input channel 9
		DI-10	%IX1.2	BOOL		Digital input channel 10
		DI-11	%IX1.3	BOOL		Digital input channel 11
		DI-12	%IX1.4	BOOL		Digital input channel 12
		DI-13	%IX1.5	BOOL		Digital input channel 13
		DI-14	%IX1.6	BOOL		Digital input channel 14
		DI-15	%IX1.7	BOOL		Digital input channel 15
		DI 16~23	%IW1	WORD		Digital input ch16 ~ ch23
		DI-16	%IX2.0	BOOL		Digital input channel 16
		DI-17	%IX2.1	BOOL		Digital input channel 17
		DI-18	%IX2.2	BOOL		Digital input channel 18
		DI-19	%IX2.3	BOOL		Digital input channel 19
		DI-20	%IX2.4	BOOL		Digital input channel 20
		DI-21	%IX2.5	BOOL		Digital input channel 21
		DI-22	%IX2.6	BOOL		Digital input channel 22
		DI-23	%IX2.7	BOOL		Digital input channel 23
		ErrorID	%IW2	WORD		Error ID currently happened in the module

5.1.6. Digital Output Modules

In this section, we are going to introduce digital output modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration, Status I/O Mapping, Status and Information.**

Status Configuration: Provide the status page for setting device. Double-click on the value column of the particular setting.



Parameter	Type	Value	Default Value	Unit	Description
Slot ID	UINT(0..31)	1	0		Slot ID
Safety Function	BOOL	FALSE	FALSE		Safety function enable/disable
Safety Value	DWORD(16#0..16#FFFFFF)	0	16#0		Safety value

Note!

If scan IO and then copy to the project is used, do not need to modify the Slot ID.

If manually insert IO into the project is used, need to configure the right Slot ID.

Status I/O Mapping: Show the I/O mapping status between variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in single bit. Set the value to "true" for switching on the channel; "false" for switching off. All channel values can represent as two WORDs.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 24 digital outputs. This will require either 24 Boolean addresses or 2 WORD (4 Byte) addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

Q = Physical Output

W = Word (16 bits)

X = Single bit

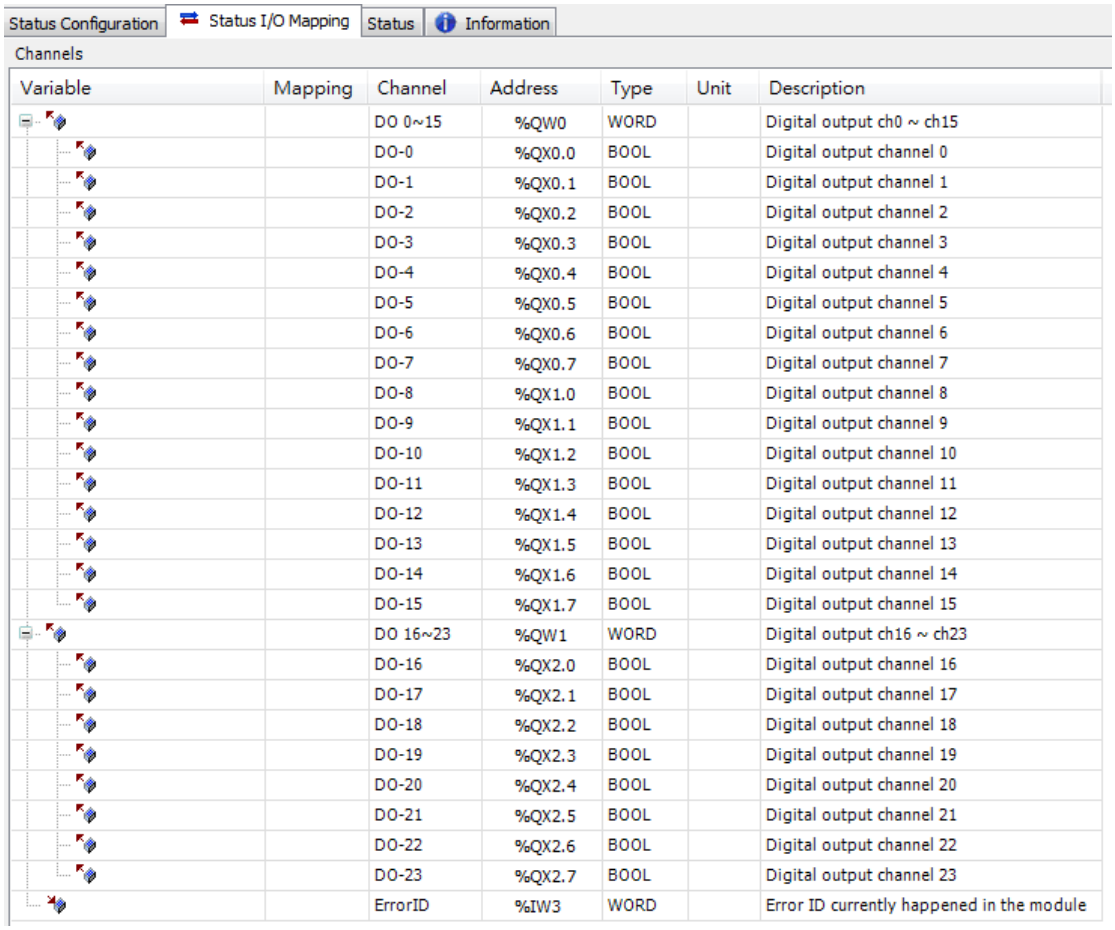
$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

Status: The reserved page.

Information: Provide the brief information to current module.



Status Configuration | Status I/O Mapping | Status | Information

Channels

Variable	Mapping	Channel	Address	Type	Unit	Description
		DO 0~15	%QW0	WORD		Digital output ch0 ~ ch15
		DO-0	%QX0.0	BOOL		Digital output channel 0
		DO-1	%QX0.1	BOOL		Digital output channel 1
		DO-2	%QX0.2	BOOL		Digital output channel 2
		DO-3	%QX0.3	BOOL		Digital output channel 3
		DO-4	%QX0.4	BOOL		Digital output channel 4
		DO-5	%QX0.5	BOOL		Digital output channel 5
		DO-6	%QX0.6	BOOL		Digital output channel 6
		DO-7	%QX0.7	BOOL		Digital output channel 7
		DO-8	%QX1.0	BOOL		Digital output channel 8
		DO-9	%QX1.1	BOOL		Digital output channel 9
		DO-10	%QX1.2	BOOL		Digital output channel 10
		DO-11	%QX1.3	BOOL		Digital output channel 11
		DO-12	%QX1.4	BOOL		Digital output channel 12
		DO-13	%QX1.5	BOOL		Digital output channel 13
		DO-14	%QX1.6	BOOL		Digital output channel 14
		DO-15	%QX1.7	BOOL		Digital output channel 15
		DO 16~23	%QW1	WORD		Digital output ch16 ~ ch23
		DO-16	%QX2.0	BOOL		Digital output channel 16
		DO-17	%QX2.1	BOOL		Digital output channel 17
		DO-18	%QX2.2	BOOL		Digital output channel 18
		DO-19	%QX2.3	BOOL		Digital output channel 19
		DO-20	%QX2.4	BOOL		Digital output channel 20
		DO-21	%QX2.5	BOOL		Digital output channel 21
		DO-22	%QX2.6	BOOL		Digital output channel 22
		DO-23	%QX2.7	BOOL		Digital output channel 23
		ErrorID	%IW3	WORD		Error ID currently happened in the module

5.1.7. Analog Input Modules

In this section, we are going to introduce analog input modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration**, **Status I/O Mapping, Status and Information.**

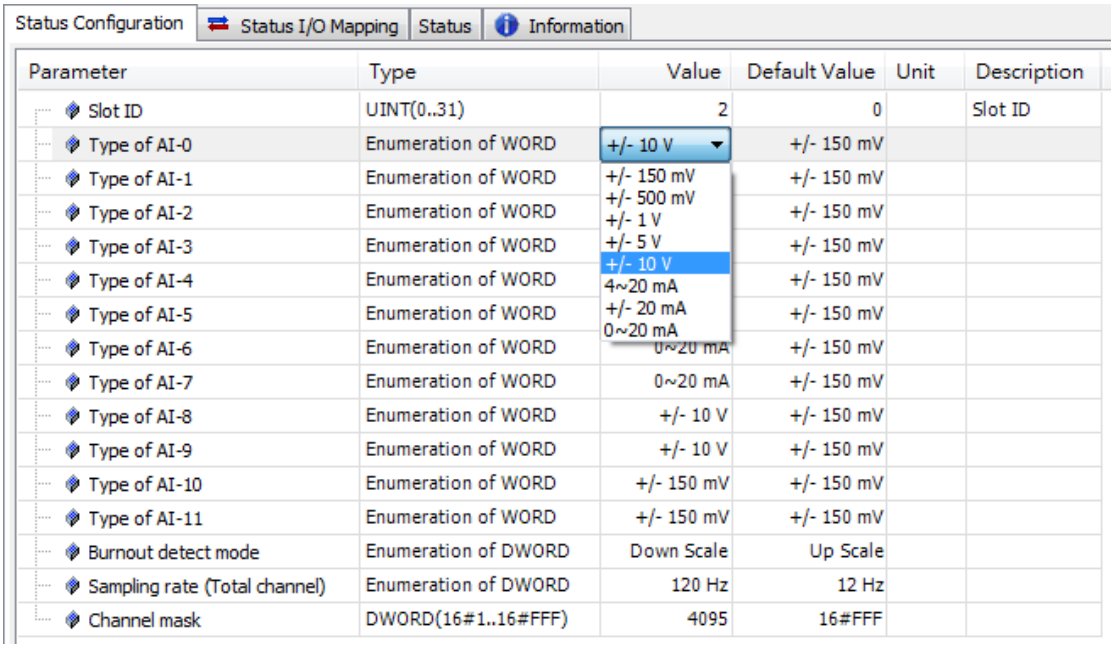
Status Configuration: Provide the channel status page for setting channel ranges.

Double-click on the value column of the particular channel.

Note!

If scan IO and then copy to the project is used, do not need to modify the Slot ID.

If manually insert IO into the project is used, need to configure the right Slot ID.



Status I/O Mapping: Show the I/O mapping status between variable to module channel.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in REAL.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 12 analog inputs. This will require 12 DWORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

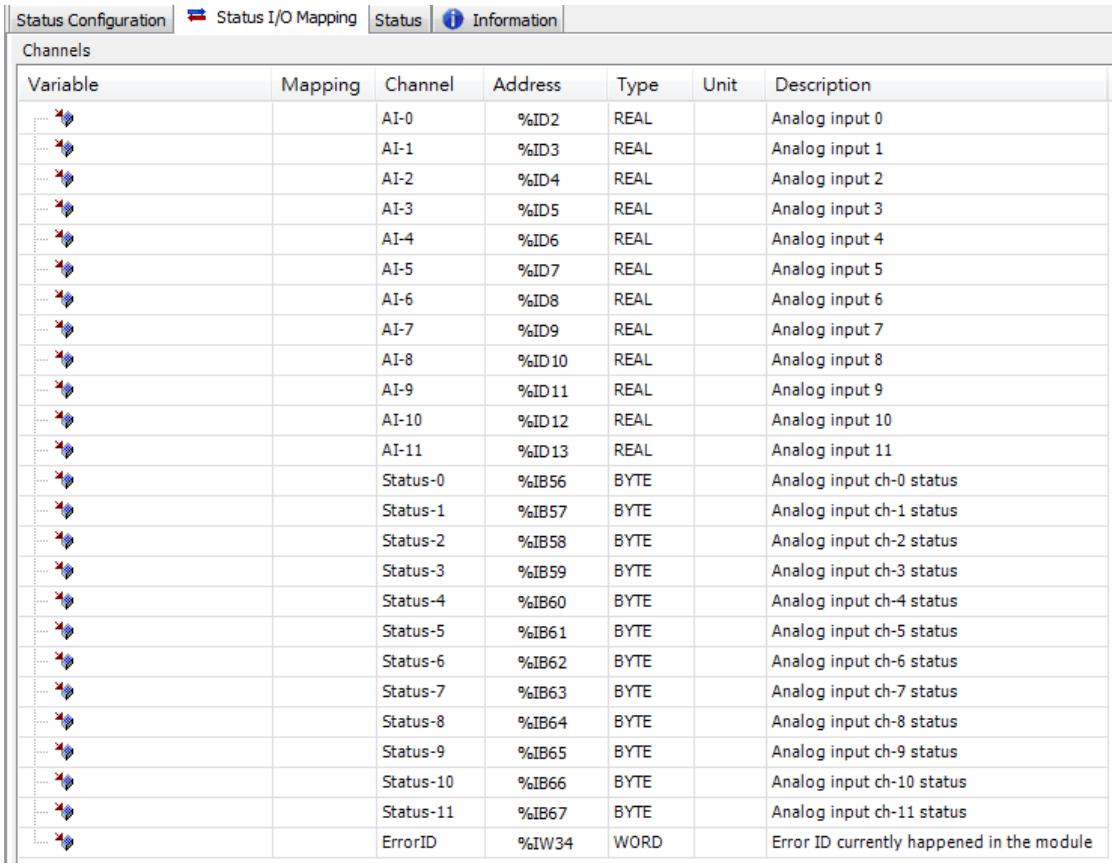
I = Physical Input

D = Double word (32 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		AI-0	%ID2	REAL		Analog input 0
		AI-1	%ID3	REAL		Analog input 1
		AI-2	%ID4	REAL		Analog input 2
		AI-3	%ID5	REAL		Analog input 3
		AI-4	%ID6	REAL		Analog input 4
		AI-5	%ID7	REAL		Analog input 5
		AI-6	%ID8	REAL		Analog input 6
		AI-7	%ID9	REAL		Analog input 7
		AI-8	%ID10	REAL		Analog input 8
		AI-9	%ID11	REAL		Analog input 9
		AI-10	%ID12	REAL		Analog input 10
		AI-11	%ID13	REAL		Analog input 11
		Status-0	%IB56	BYTE		Analog input ch-0 status
		Status-1	%IB57	BYTE		Analog input ch-1 status
		Status-2	%IB58	BYTE		Analog input ch-2 status
		Status-3	%IB59	BYTE		Analog input ch-3 status
		Status-4	%IB60	BYTE		Analog input ch-4 status
		Status-5	%IB61	BYTE		Analog input ch-5 status
		Status-6	%IB62	BYTE		Analog input ch-6 status
		Status-7	%IB63	BYTE		Analog input ch-7 status
		Status-8	%IB64	BYTE		Analog input ch-8 status
		Status-9	%IB65	BYTE		Analog input ch-9 status
		Status-10	%IB66	BYTE		Analog input ch-10 status
		Status-11	%IB67	BYTE		Analog input ch-11 status
		ErrorID	%IW34	WORD		Error ID currently happened in the module

Status: The reserved page.

Information: Provide the brief information for current module.

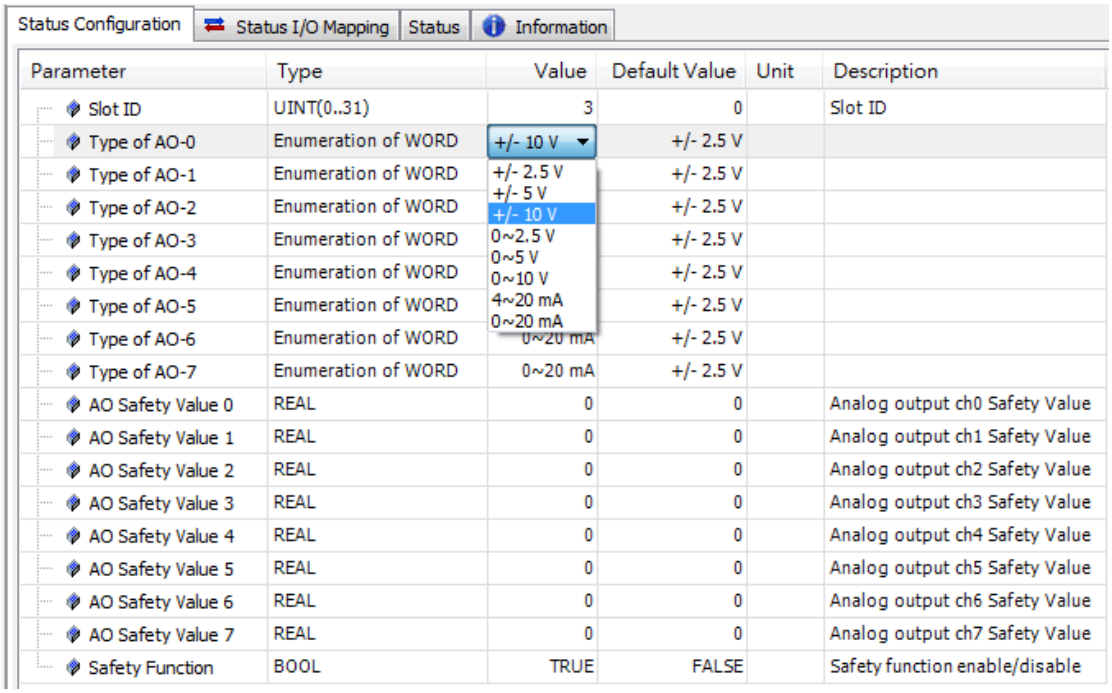
5.1.8. Analog Output Modules

In this section, we are going to introduce analog output modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration**, **Status I/O Mapping, Status and Information.**

Status Configuration: Provide the channel status page for setting channel ranges.

Double-click on the value column of the particular channel.



Note!

If scan IO and then copy to the project is used, do not need to modify the Slot ID.

If manually insert IO into the project is used, need to configure the right Slot ID.

Status I/O Mapping: Show the I/O mapping status between local variable to module channel.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in REAL.

For detailed variable mapping information, see chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 analog outputs. This will require 8 DWORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

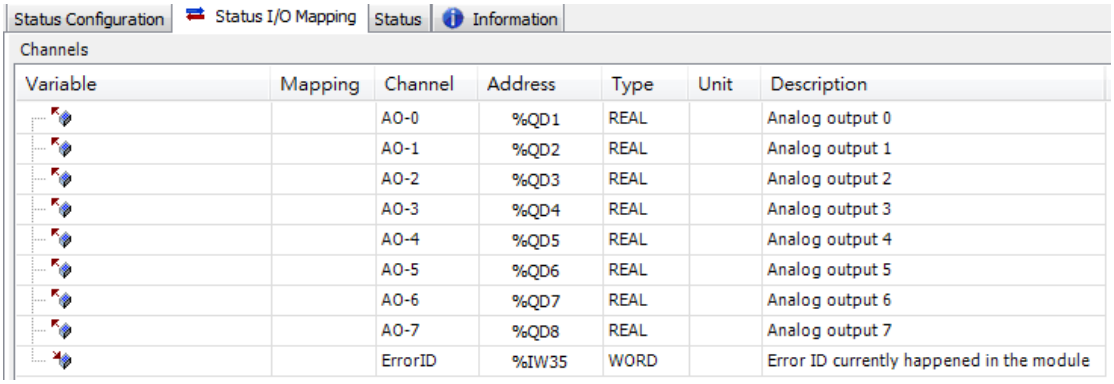
Q = Physical Output

D = Double word (32 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		AO-0	%QD1	REAL		Analog output 0
		AO-1	%QD2	REAL		Analog output 1
		AO-2	%QD3	REAL		Analog output 2
		AO-3	%QD4	REAL		Analog output 3
		AO-4	%QD5	REAL		Analog output 4
		AO-5	%QD6	REAL		Analog output 5
		AO-6	%QD7	REAL		Analog output 6
		AO-7	%QD8	REAL		Analog output 7
		ErrorID	%IW35	WORD		Error ID currently happened in the module

Status: The reserved page.

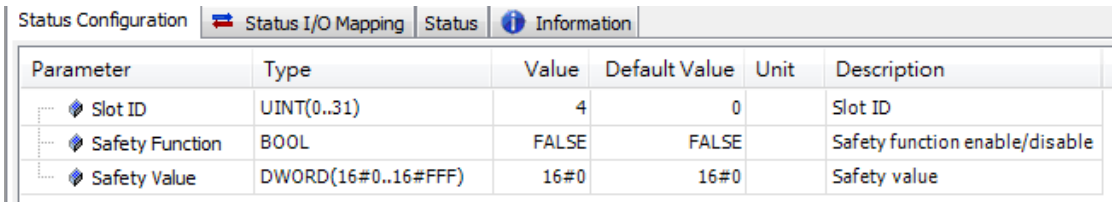
Information: Provide the brief information for current module.

5.1.9. Relay Output Modules

In this section, we are going to introduce relay output modules.

The **Module editor** opens by double clicking the device name in the device tree. It consists of four tab pages, that is, **Status Configuration**, **Status I/O Mapping, Status and Information.**

Status Configuration: Provide the status page for setting device. Double-click on the value column of the particular setting.



Note!

If scan IO and then copy to the project is used, do not need to modify the Slot ID.

If manually insert IO into the project is used, need to configure the right Slot ID.

Status I/O Mapping: Show the I/O mapping status between local variable to module channel. It consists of seven columns.

Mapping: The mapping status of each variable.

Note!

There are two categories of variables: **Channel values** and **Error ID**.

Channel values: The data type of each channel is in single bit. Set the value to "true" for switching on the channel; "false" for switching off. All channel values can represent as one word.

For detailed variable mapping information, see Chapter 4.2.

Error ID: This variable holds the status of I/O module and its data type is in Word (16 Bits). Get module error ID by mapping the last variable in table. For detailed error ID information, see Chapter 5.3.

Address: The starting physical address of the variables for this I/O group. The board shown below has 12 relay outputs. This will require either 12 Boolean addresses or 1 WORD address.

Note!

Meaning of address expression:

% = Directly Mapped variable

Q = Physical Output

W = Word (16 bits)

X = Single bit

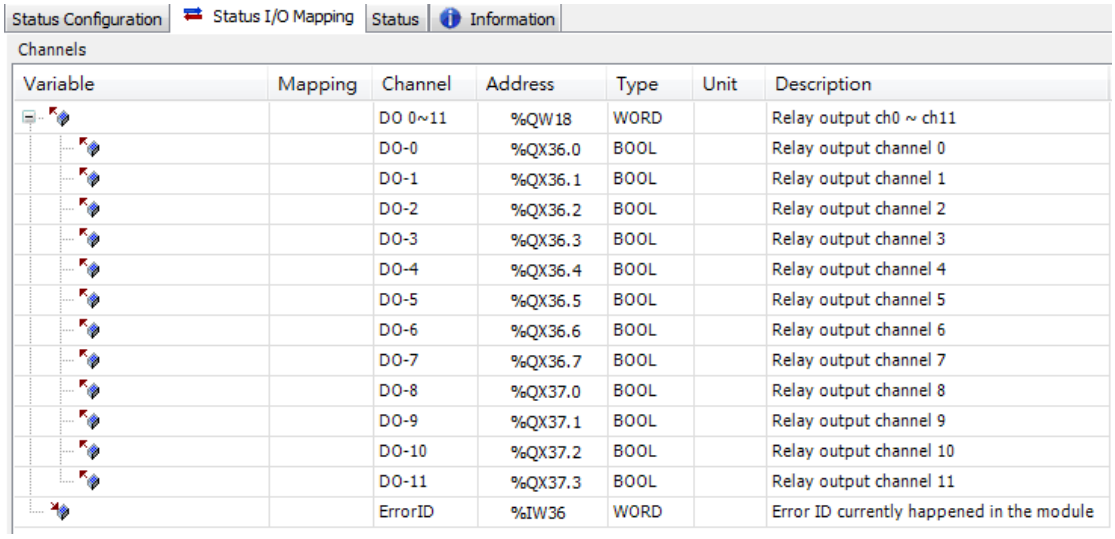
$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

Status: The reserved page.

Information: Provide the brief information to current module.



Variable	Mapping	Channel	Address	Type	Unit	Description
		DO 0~11	%QW18	WORD		Relay output ch0 ~ ch11
		DO-0	%QX36.0	BOOL		Relay output channel 0
		DO-1	%QX36.1	BOOL		Relay output channel 1
		DO-2	%QX36.2	BOOL		Relay output channel 2
		DO-3	%QX36.3	BOOL		Relay output channel 3
		DO-4	%QX36.4	BOOL		Relay output channel 4
		DO-5	%QX36.5	BOOL		Relay output channel 5
		DO-6	%QX36.6	BOOL		Relay output channel 6
		DO-7	%QX36.7	BOOL		Relay output channel 7
		DO-8	%QX37.0	BOOL		Relay output channel 8
		DO-9	%QX37.1	BOOL		Relay output channel 9
		DO-10	%QX37.2	BOOL		Relay output channel 10
		DO-11	%QX37.3	BOOL		Relay output channel 11
		ErrorID	%IW36	WORD		Error ID currently happened in the module

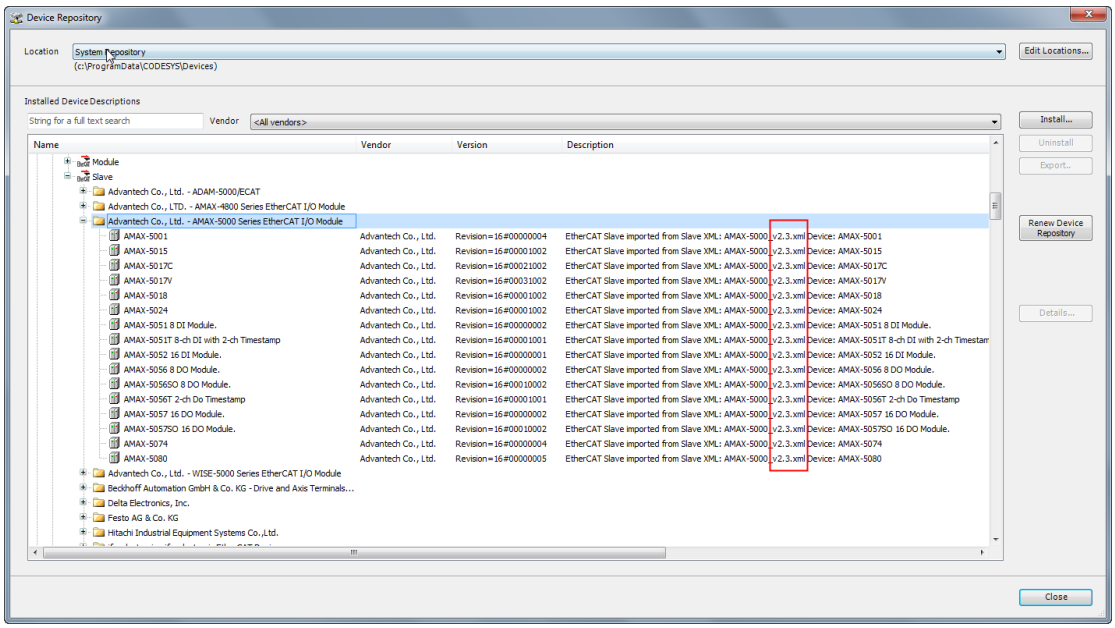
5.2. AMAX-5000 IO Modules

5.2.1. Scan AMAX I/O Modules into CODESYS

AMAX-5000 series EtherCAT Slice IO modules are standard EtherCAT IO modules, the standard process to equip EtherCAT fieldbus has described in Chapter 6.5.

Before connecting to slaves by using EtherCAT client, please install the related EtherCAT XML device description files (*.xml) first. Please get this file from Advantech support website. Or contact your window of technical support.

If user follows the Chapter 2.2 to install Advantech Add-on Package, the latest device description file would be well installed. Please check the version in the dialog window of **Device Repository (Tool → Device Repository)**

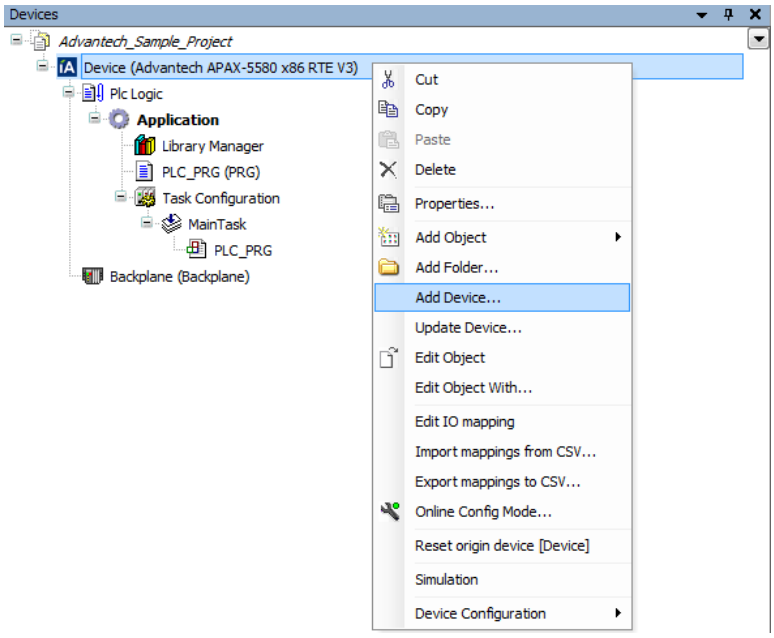


If user needs to install another version of description file, please click “Install” button and select the related file you want to install.

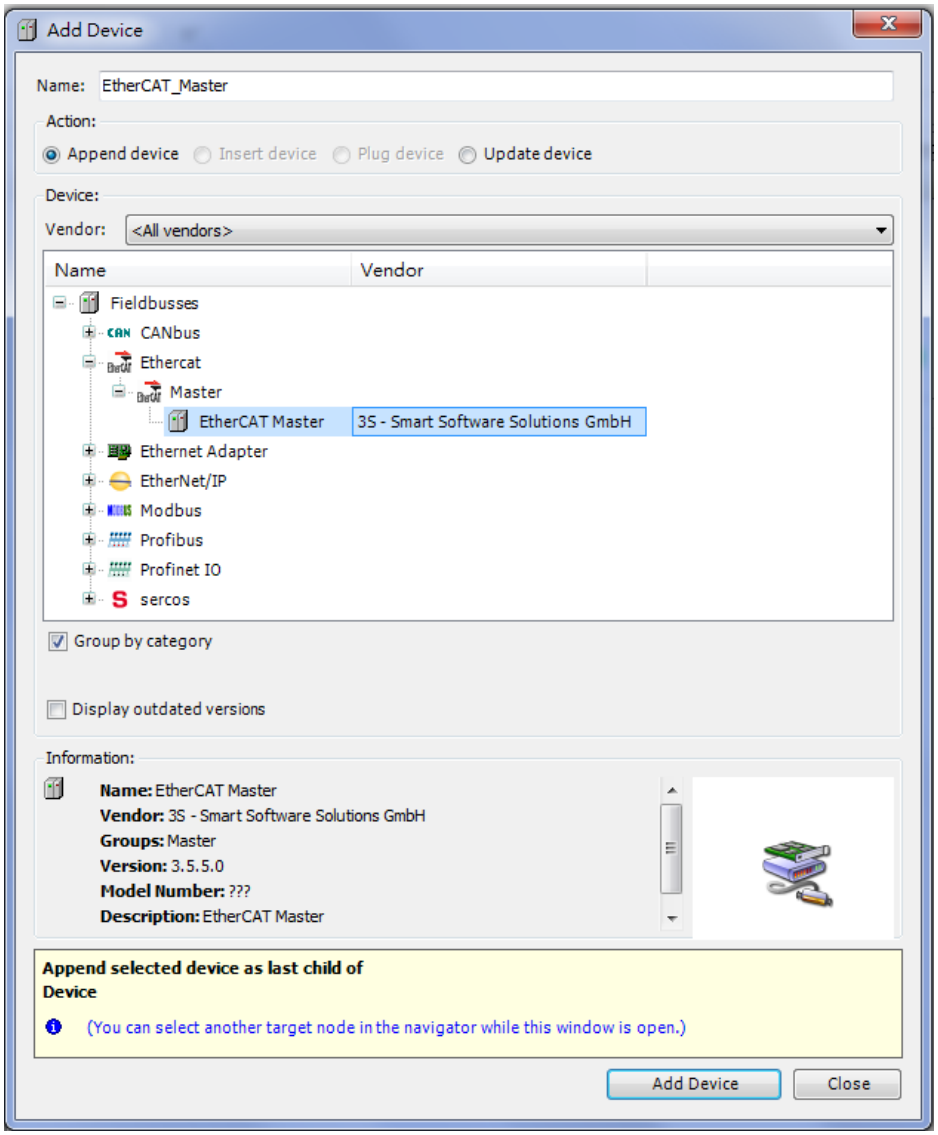
After installed successfully, the following information will be shown: “Device xxxx installed to device repository”.

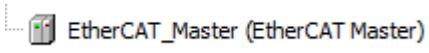
Before scan the device, user has to connect to specified Advantech X86 RTE platform. Please refer to Chapter 3.4 for detail.

To scan the EtherCAT device, user has to add EtherCAT Master first. In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog.



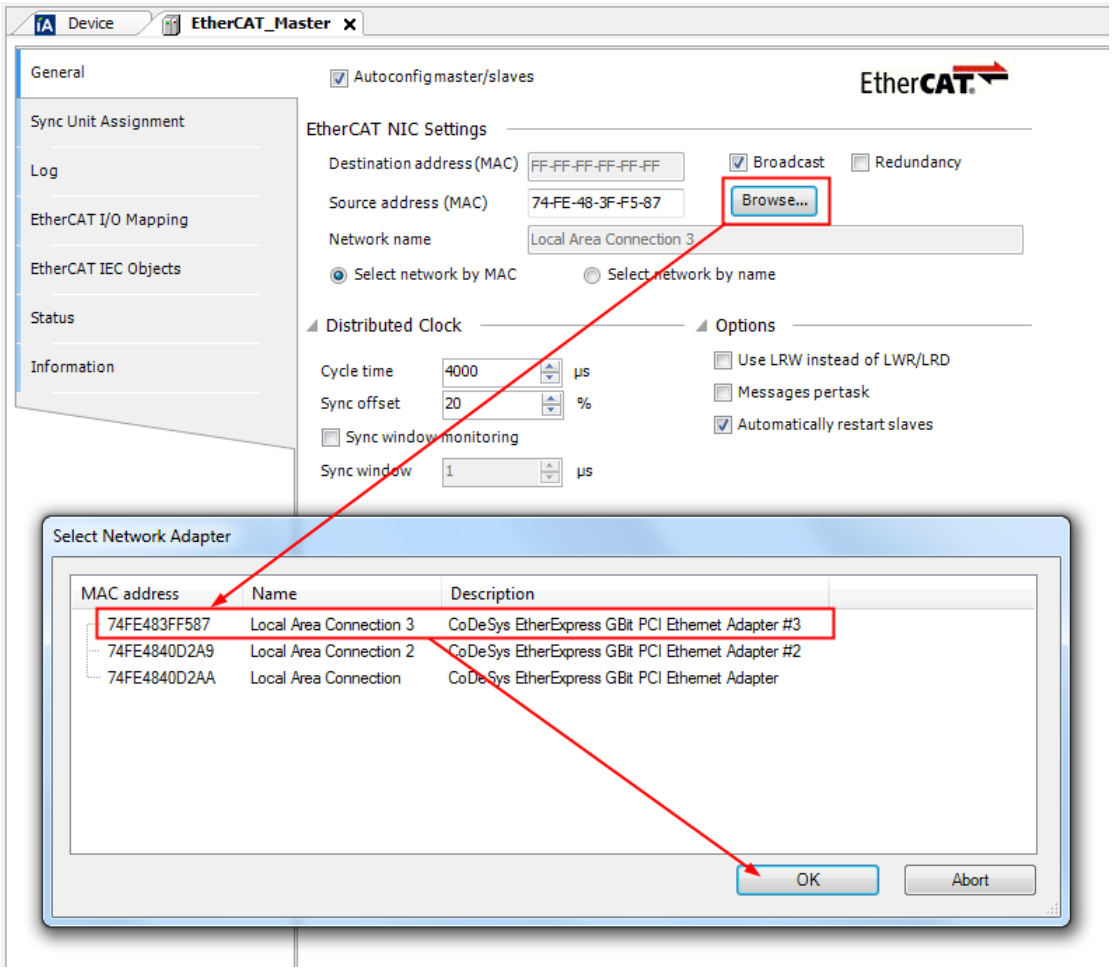
Choose **EtherCAT Master** in the **EtherCAT/Master** option and click **Add Device** to proceed and then press Close to close the device dialog.



Now, you'll see EtherCAT Master 

in the device tree.

Double-click the EtherCAT Master icon to set configuration. Click **Browse** and then select proper Ethernet Adapter.

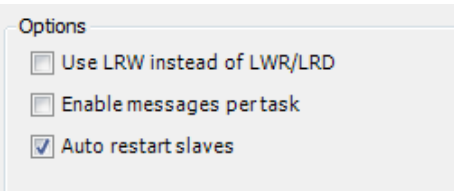


Note!

(1) Different platform would have different result of Network Adapter, for AMAX-5580 LAN1 and LAN3 are recommended to set as EtherCAT Master Port. And the MAC address is listed on the side of AMAX-5580 as reference.

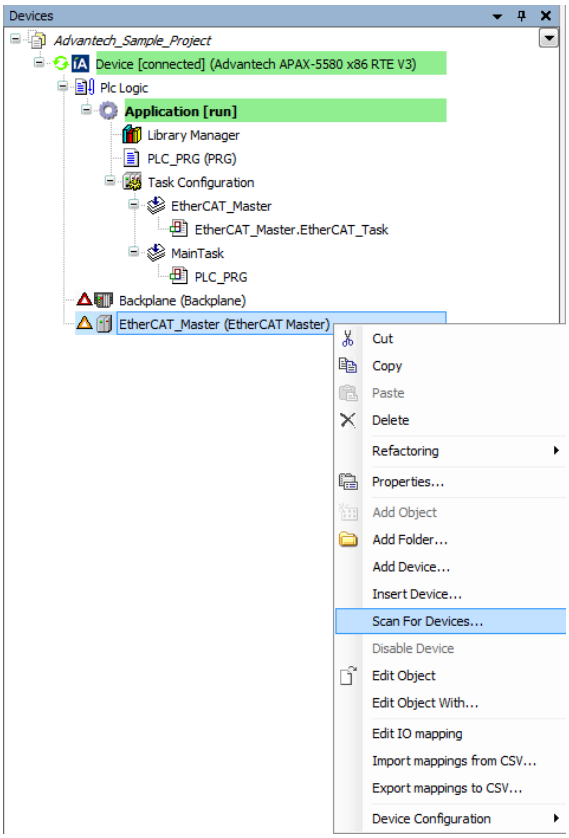
(2) For APAX-5580, only LAN2 can be selected as EtherCAT Master Port

(3) “Auto restart slaves” is suggested to be clicked.

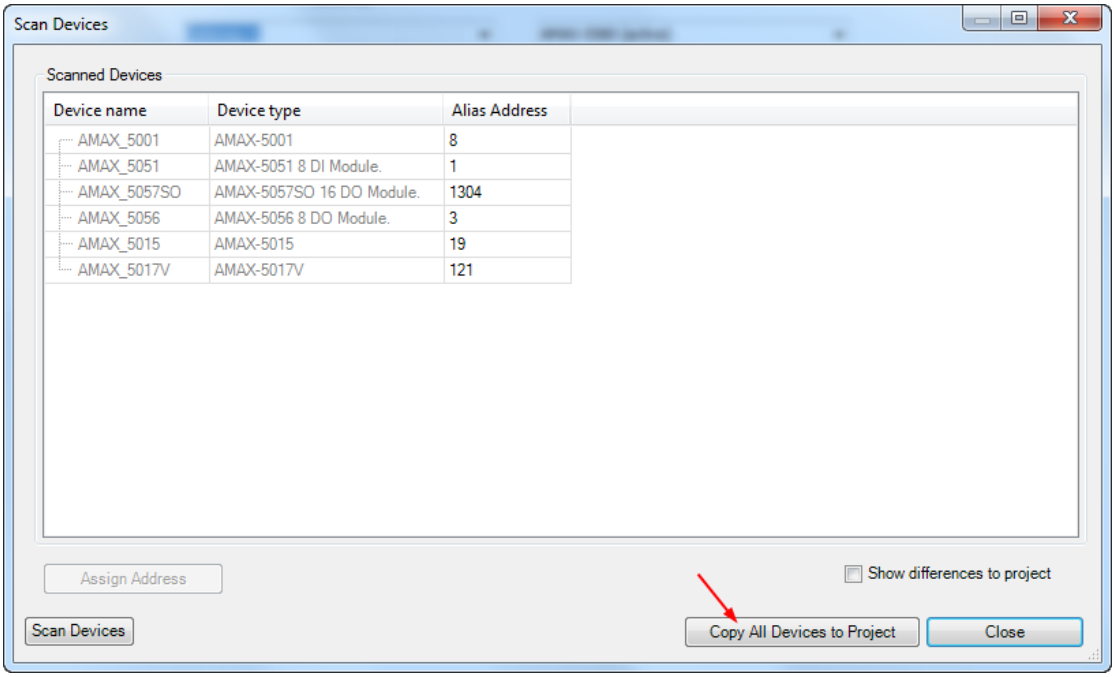


At the first scan, at least **once a login (and running)** must have been done. Otherwise, the Advantech X86 RTE platform must be running before a scan.

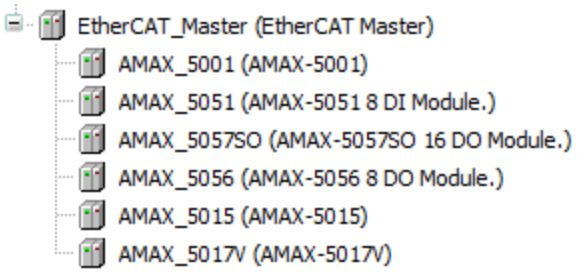
Choose the **EtherCAT_Master** and right click **Scan For Devices** in context menu.



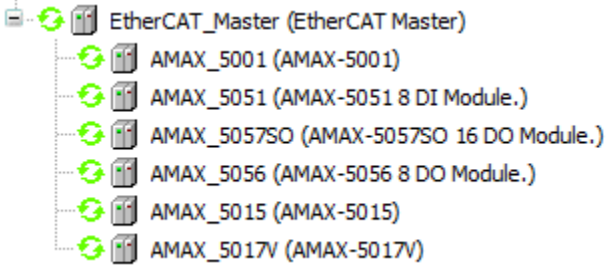
A list of all devices and modules are found during the last scan. Select the specified one device and then copy to the project. Or copy all listed devices to the project.



Now, you'll see the devices copied to the project under the **EtherCAT_Master**.



After completely finish device configuration, it is necessary to login again.



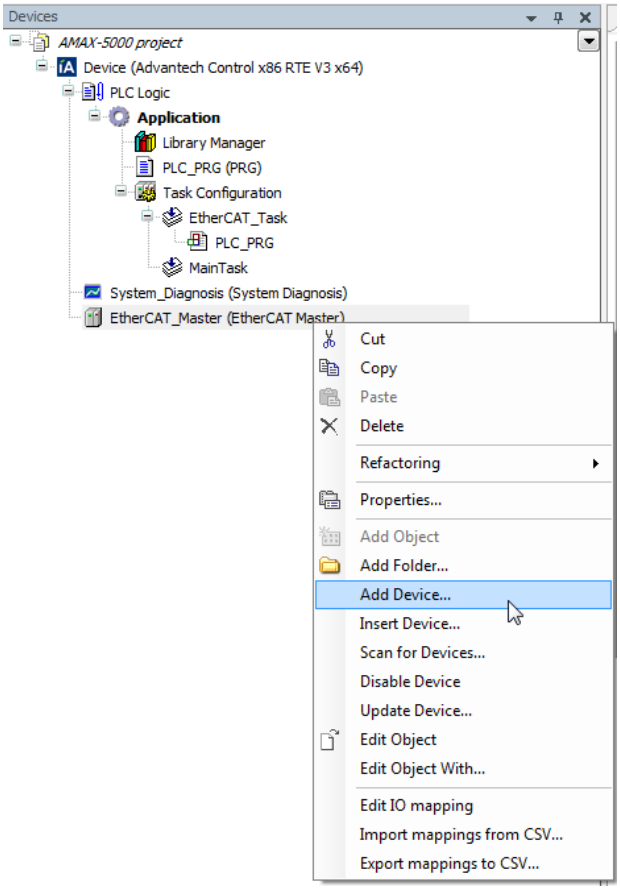
Note!

You can remove the existing device by click **Delete** in context menu.

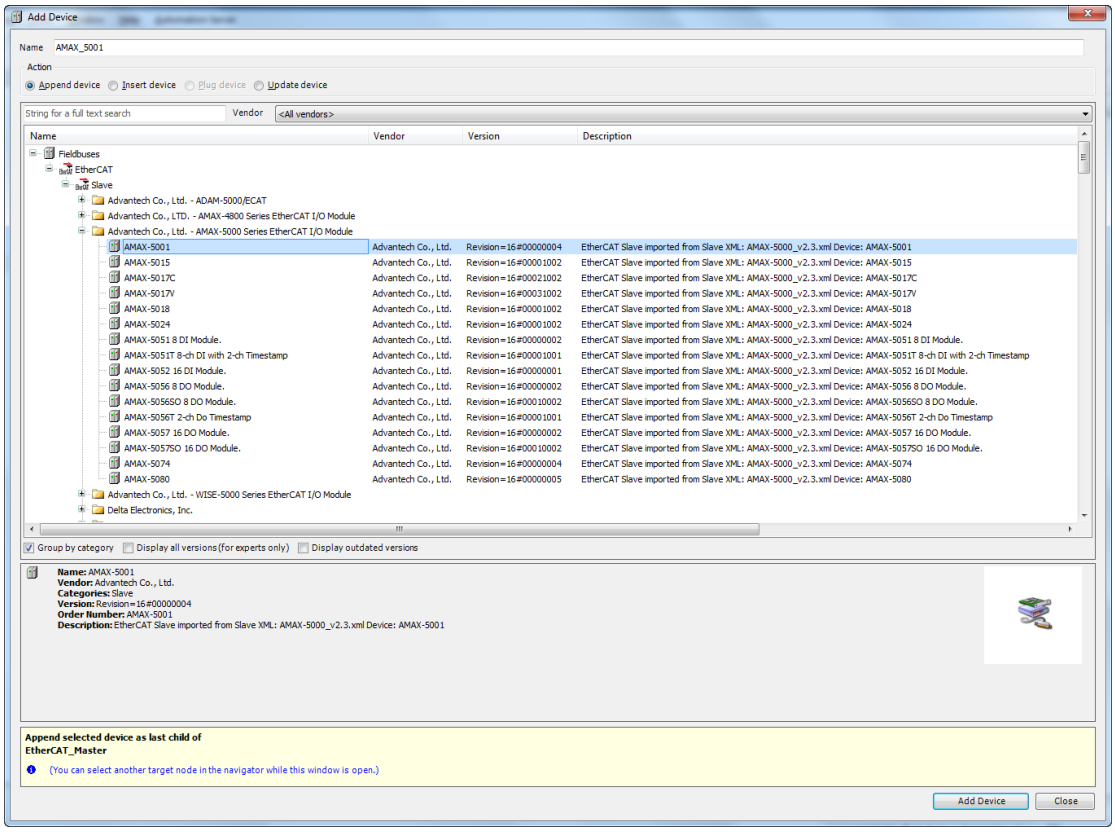
5.2.2. Insert AMAX I/O Modules into CODESYS

If user do not have the I/O modules on line, user can add and configure Advantech AMAX-5000 I/O modules as objects in the device tree in off-line mode.

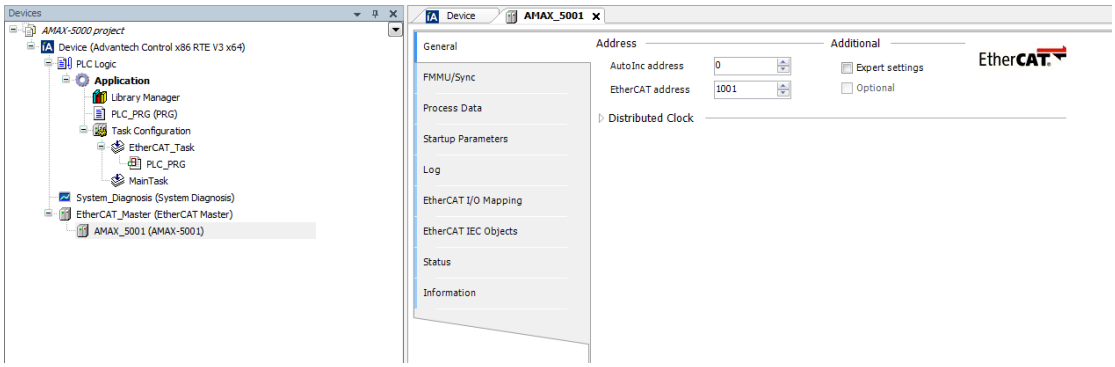
Choose the **EtherCAT Master** in the tree view and click **Add Device** in the context manual.



It will open the **Add Device dialog**, where you can choose one of available devices. Click **Add Device** to proceed and then press **Close** to close the device dialog.

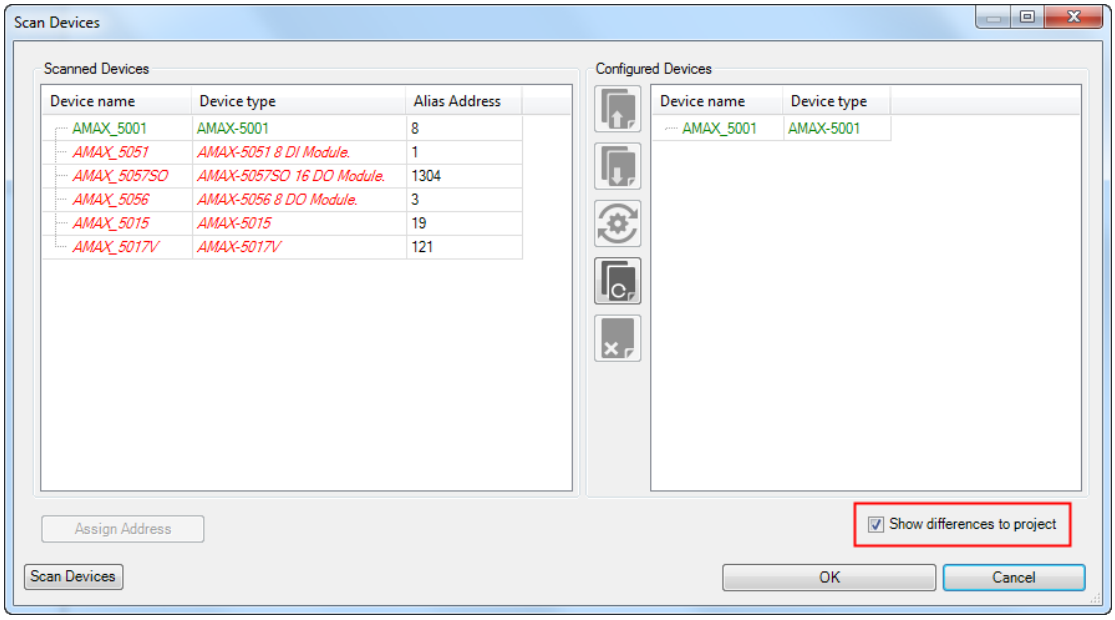


Then the specified IO module will be added into the Backplane.



Note!

If user inserts I/O modules manually, user has to configure the AMAX-5000 I/O modules to map to the same sequence physically. And suggest user do scanning again and checking the box of **Show differences to project** to compare and adjust the I/O system.


Scan Devices
Scanned Devices
Device name
Device type
Alias Address
AMAX_5001
AMAX-5001
8
AMAX_5051
AMAX-5051 8 DI Module.
1
AMAX_5057SO
AMAX-5057SO 16 DO Module.
1304
AMAX_5056
AMAX-5056 8 DO Module.
3
AMAX_5015
AMAX-5015
19
AMAX_5017V
AMAX-5017V
121
Configured Devices
Device name
Device type
AMAX_5001
AMAX-5001
Assign Address
Show differences to project
Scan Devices
OK
Cancel

5.2.3. Map Variables to I/O Modules

In this section, we want to discuss how to map variable of program to Advantech I/O modules. For more details on creating a new program please refer to Chapter 3.

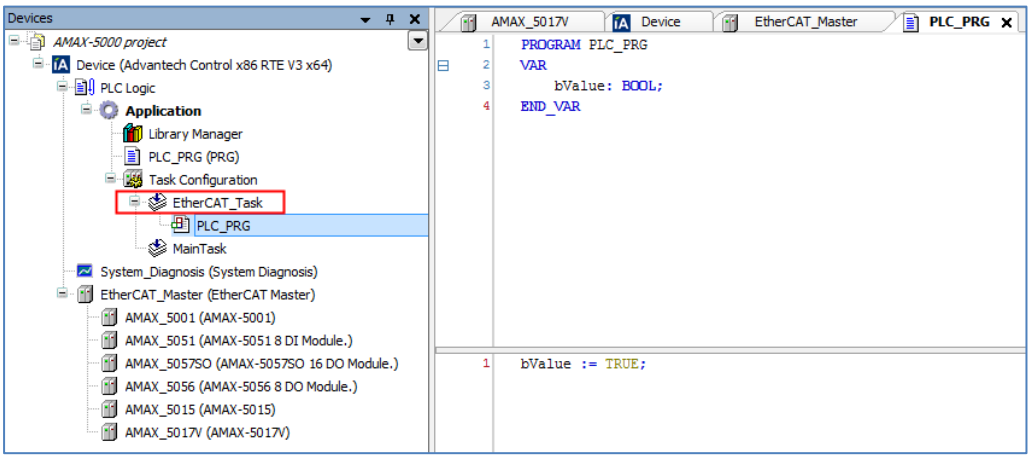
Here, we declare *bValue* in declaration part and set true in body part.

```
PROGRAM PLC_PRG
VAR
    bValue: BOOL;
END_VAR
```

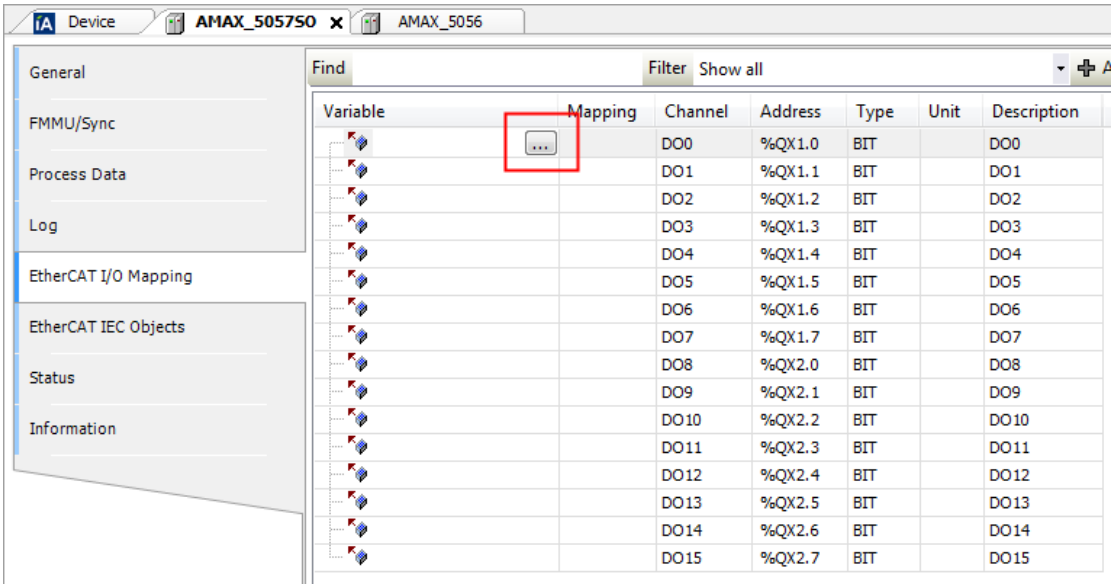
```
bValue:= true;
```

Note!

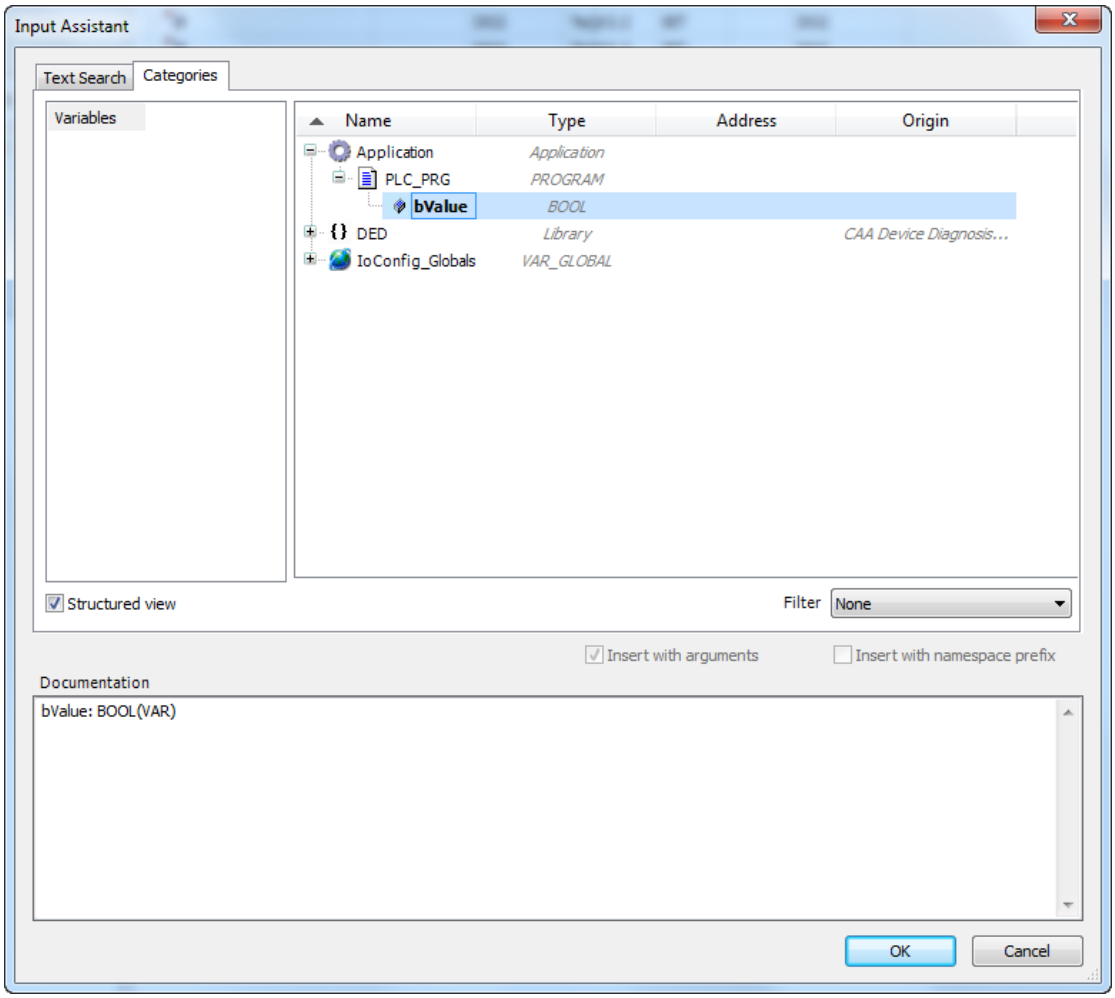
If the program will be used to access the EtherCAT IO, the program need to be under the **EtherCAT_Task**, this task will synchronize with the EtherCAT Master



Open **Device Editor** by double clicking the device name in the device tree. Double-click on the variable column and choose mapping variable by clicking the button [...]. In this example, we try to map the variable (*bValue*) to channel 0, so we double-click on the first row of variable column.



It will open the **Input Assistant Dialog**, where you can choose one of available variables for the current digital output channel.

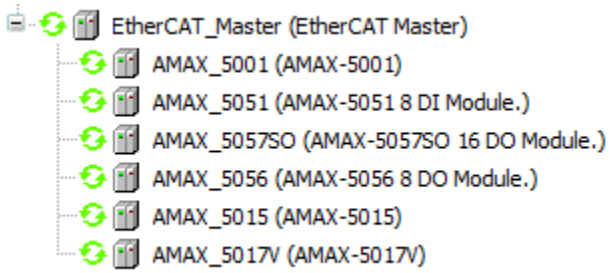


Now, we can download the application by performing command **Login** and then performing command **Start**. The channel-0 of I/O module will be lit up.

Variable	Mapping	Channel	Address	Type	Current Value	Prepared Value	Unit	Description
Application.PLC_PRG.bValue		DO0	~~%QX3.0~~	BIT	TRUE			DO0
		DO1	%QX3.1	BIT	FALSE			DO1
		DO2	%QX3.2	BIT	FALSE			DO2
		DO3	%QX3.3	BIT	FALSE			DO3
		DO4	%QX3.4	BIT	FALSE			DO4
		DO5	%QX3.5	BIT	FALSE			DO5
		DO6	%QX3.6	BIT	FALSE			DO6
		DO7	%QX3.7	BIT	FALSE			DO7

Note!

If the Advantech modules are correctly configured, it will show a green circle icon next to the device name in the device tree. If it shows a red triangle , see Chapter 8 for troubleshooting.



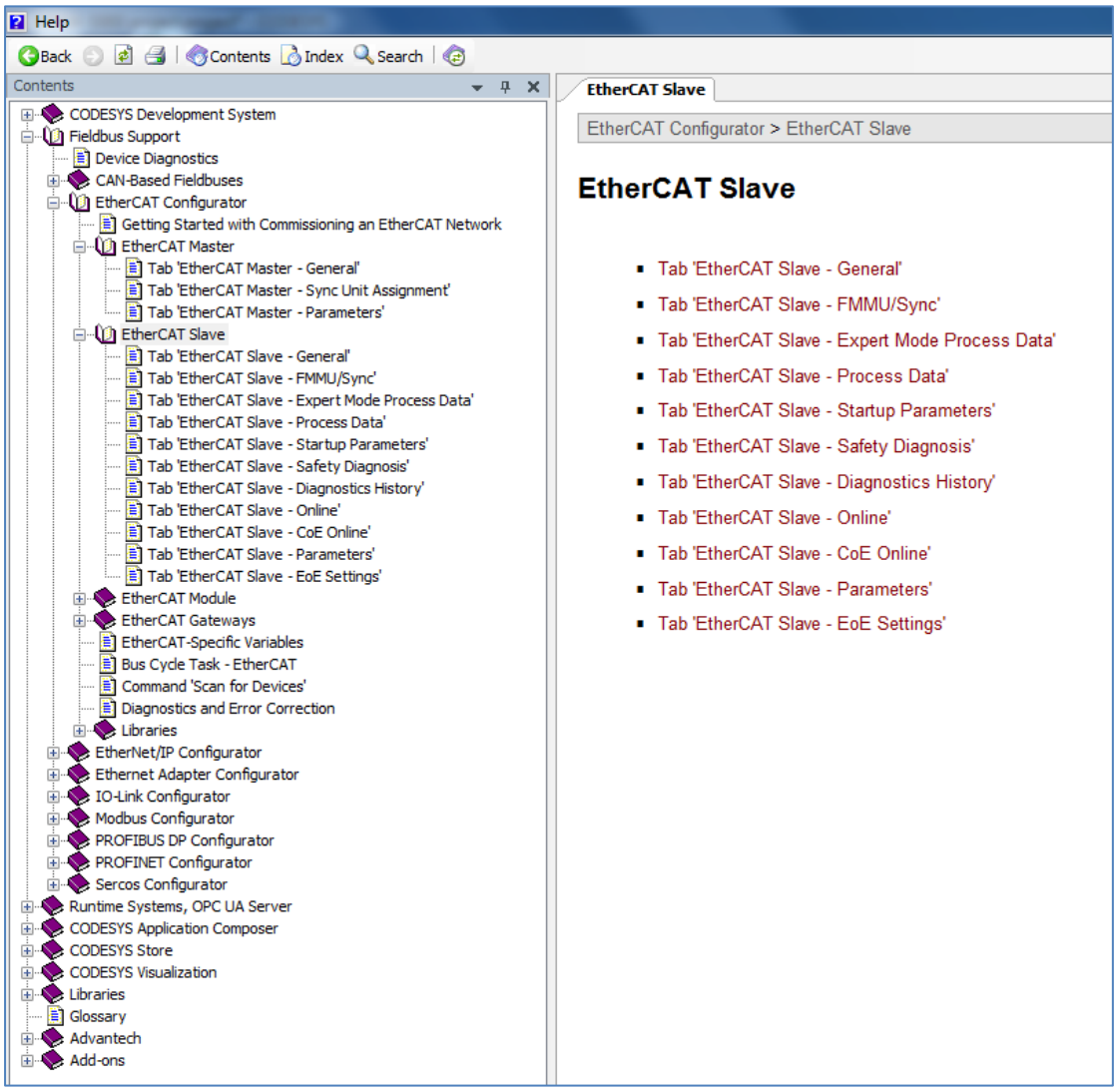
5.2.4. Support List

Advantech provides 16 types of AMAX-5000 I/O modules for various applications so far. Following table is the I/O modules support list. In the following section, we will introduce I/O modules according to their types.

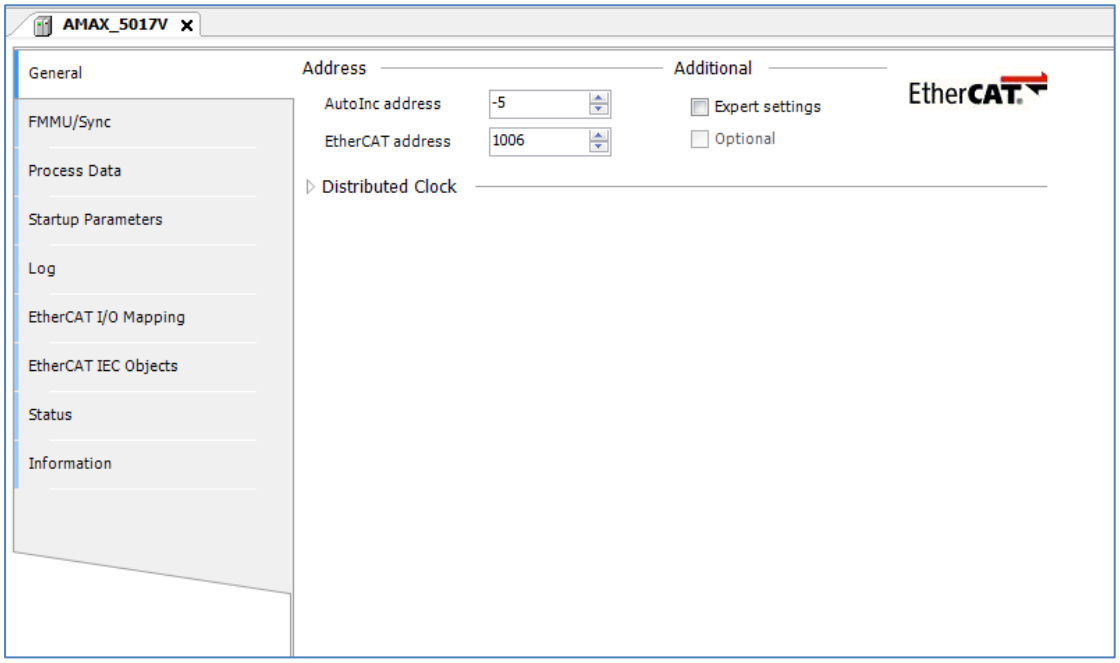
Module	Name	Specification	Reference
Power Input & Coupler	AMAX-5001	Power Input with 4-ch DI	Isolated
	AMAX-5074	Power Input & Coupler with ID	Isolated
Analog Input	AMAX-5015	4-ch RTD Input	Isolated
	AMAX-5017C	6-ch AI /Current Type	Isolated
	AMAX-5017V	6-ch AI /Voltage Type	Isolated
	AMAX-5018	6-ch Thermocouple Input	Isolated
Analog Output	AMAX-5024	4-ch AO	Isolated
Digital Input	AMAX-5051	8-ch DI w/LED	Isolated
	AMAX-5051T	2-ch DI w/Timestamp, 6-ch DI	Isolated
	AMAX-5052	16-ch DI w/LED	Isolated
Digital Output	AMAX-5056	8-ch DO (sink) w/LED	Isolated
	AMAX-5056SO	8-ch DO (source) w/LED	Isolated
	AMAX-5056T	2-ch DO (sink) w/Timestamp	Isolated
	AMAX-5057	16-ch DO (sink) w/LED	Isolated
	AMAX-5057SO	16-ch DO (source) w/LED	Isolated
Counter	AMAX-5080	2-ch Counter/Encoder 32-bit	Isolated

5.2.5. EtherCAT Slave Modules Configuration

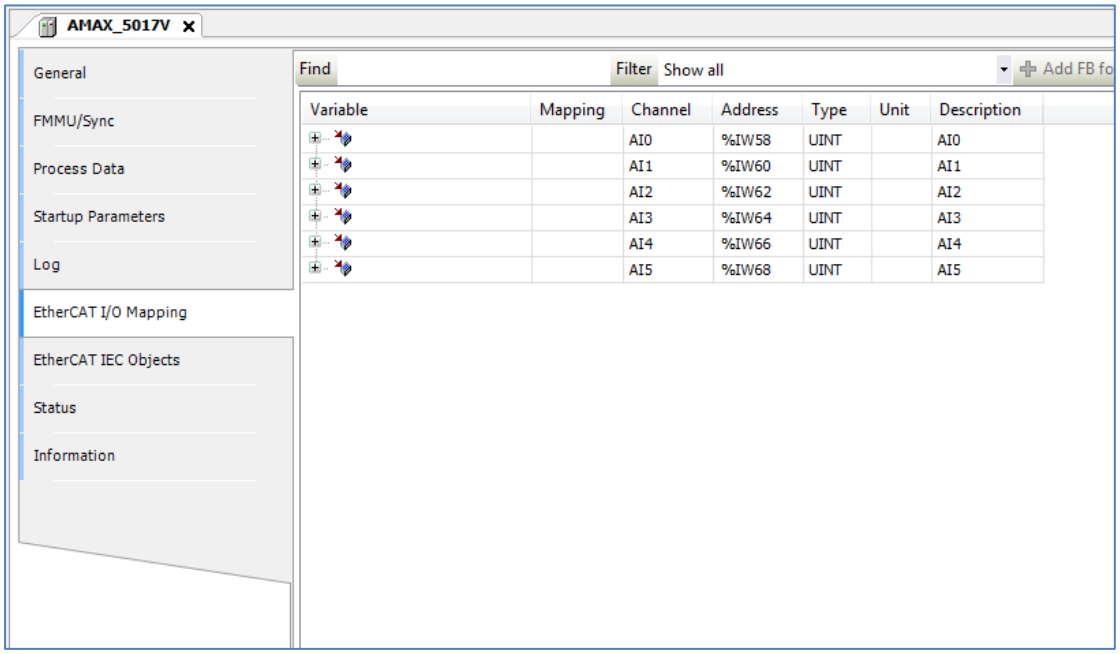
Advantech AMAX-5000 IO modules follow the standard EtherCAT slave stack and can be identified as standard EtherCAT slave modules under CODESYS. This section is to introduce the general setting for EtherCAT slave modules. And user can also refer to the CODESYS on line help for more detail.



Double click the target EtherCAT slave object under the tree of EtherCAT Master, the device editor will be shown as below:

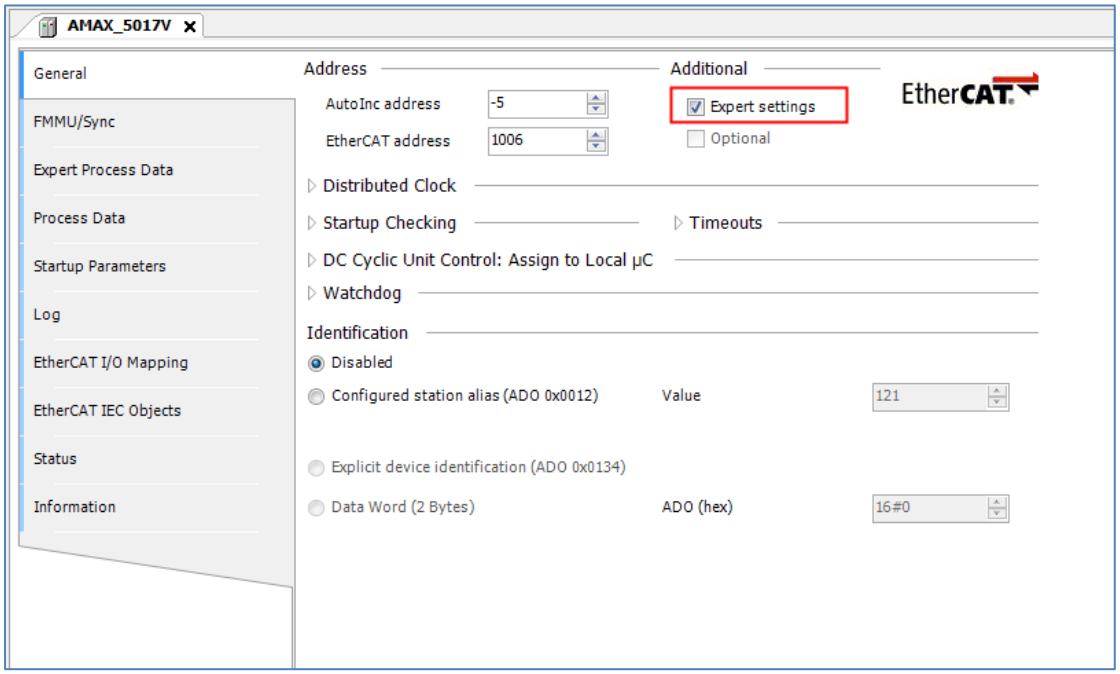


In this dialog window, user can easily identify the EtherCAT address has been assigned automatically. And all the information is from device description file. User can also unfold the **Distributed Clock** option to change the related setting. Usually user can directly go the **EtherCAT I/O Mapping** tab to map variables to the physical I/O.

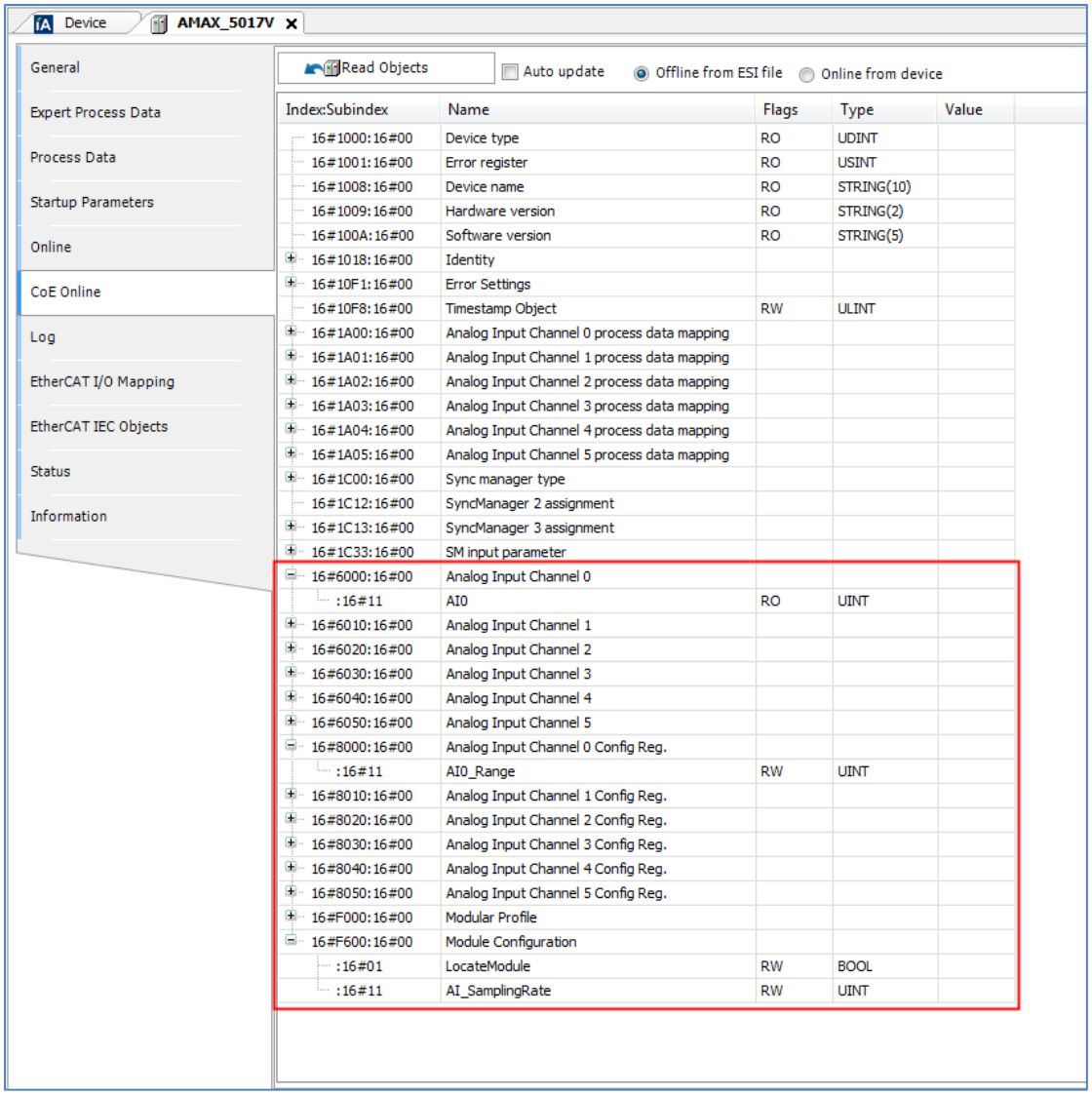


If user needs additional setting for the startup checking and time monitoring, the **Expert**

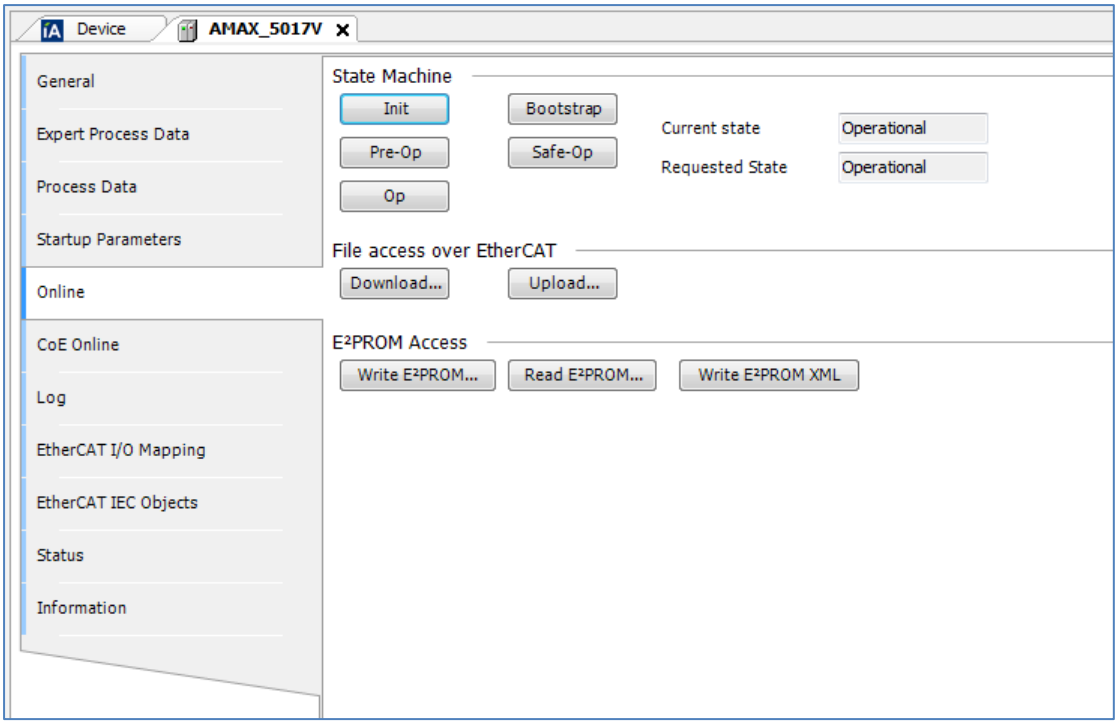
Process Data option can be selected. However, for standard applications, auto-configuration is sufficient.



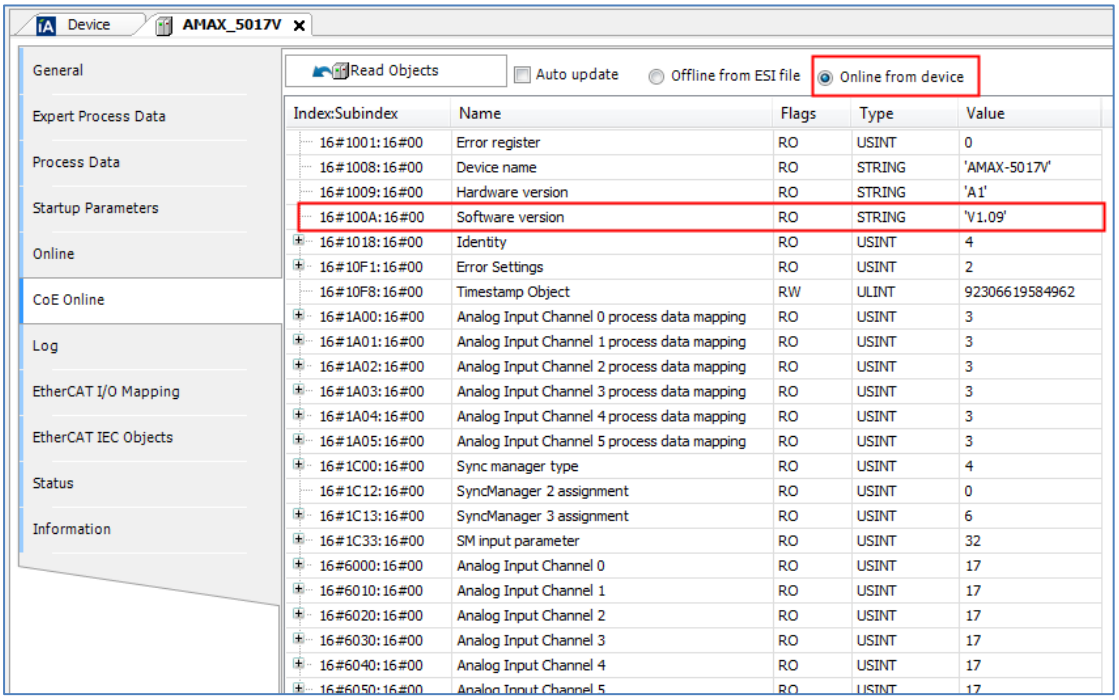
Some AMAX-5000 modules support various working modes, and need user to specify the working parameters. For example, setting the range code or sample rate for analogue input modules. These parameters can only be set while the module is on-line. To achieve this, please click the **Login** and check the **Expert settings** option, and then check the **CoE Online** tab as show below. For each function of individual module, please refer to AMAX-5000 user's manual.



In the **Online** tab, user can change the State Machine of modules and update the firmware. To update the firmware, user has to click the **Bootstrap** button to change the state machine to be **Bootstrap** mode. Then click the Download button and select the target file to download. To prevent the human error, user has the enter password to proceed the download. The default password (Hex) is **"55aa55aa".** After finishing the download, please click **Init** button to switch the State Machine to Initial mode, the firmware will be updated automatically in the device. Please wait for 5 seconds and reset the device by **Online → Reset Origin**. The device will come back to **operational** mode.



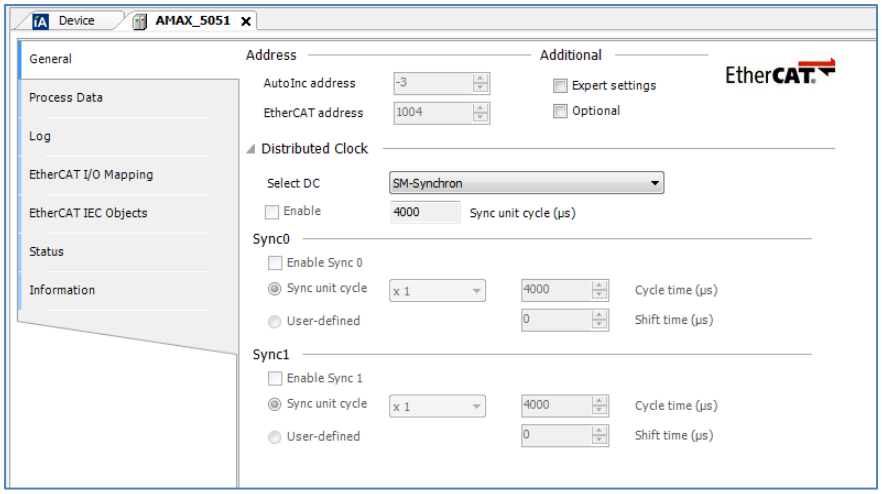
After the firmware update, please go back to CoE Online to check if the correct version is updated.

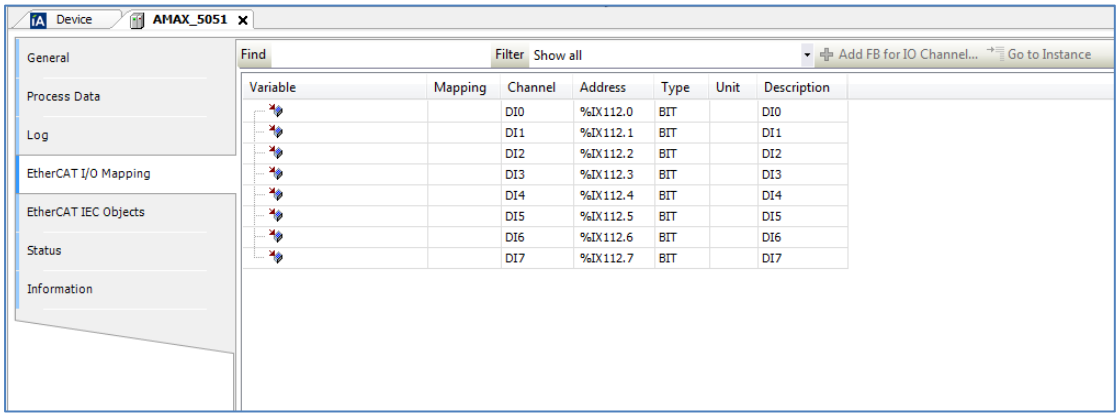


5.2.6. Digital Input Modules

In this section, we are going to introduce AMAX-5000 digital input modules, and take AMAX-5051 as example. The **device editor** dialog window can be opened by double clicking the device name in the device tree.

In the **General tab**, user can easily identify the EtherCAT address has been assigned automatically. And all the information is from device description file. User can also unfold the **Distributed Clock** option to change the related setting. Usually user can directly go the **EtherCAT I/O Mapping** tab to map variables to the physical I/O.





Variable: Double click the column and select the proper variable for mapping.

Mapping: The mapping status of each variable.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 digital inputs. This will require either 8 Boolean addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

I = Physical Input

X = Single bit

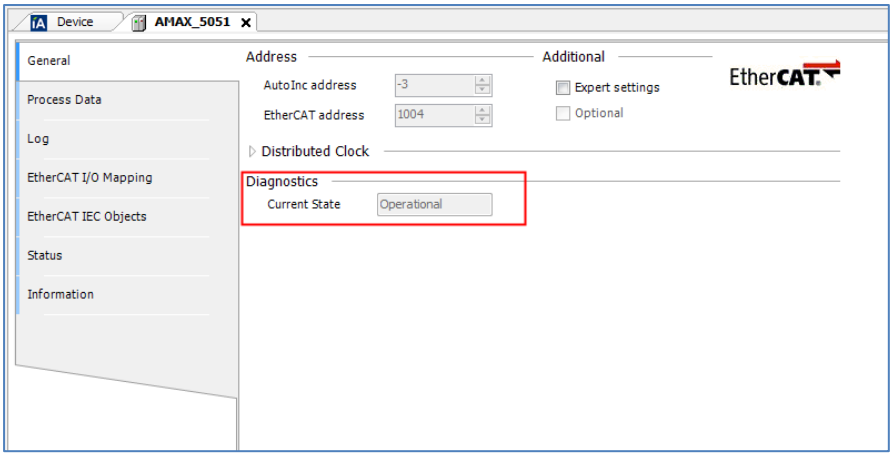
$(N1). $(N2) = Starting address. The first number means the starting byte; the second number means the starting bit.

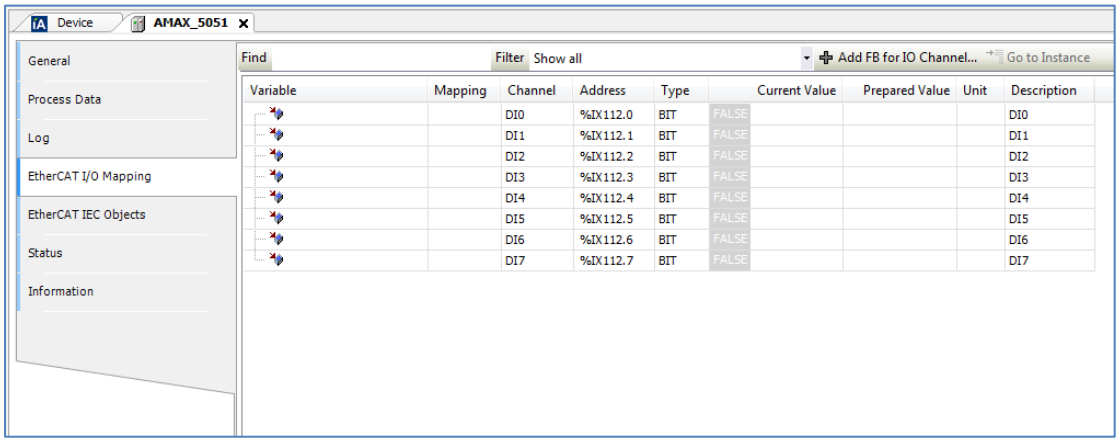
Type: The data type of each variable.

Unit: Reserved Column

Description: The description of each variable.

User can also **Login** to be on-line and check the status of modules.

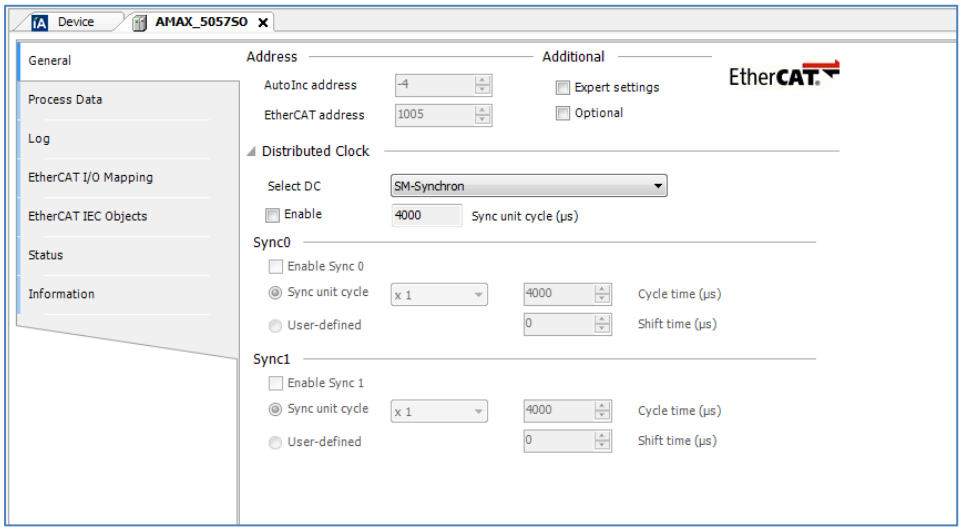


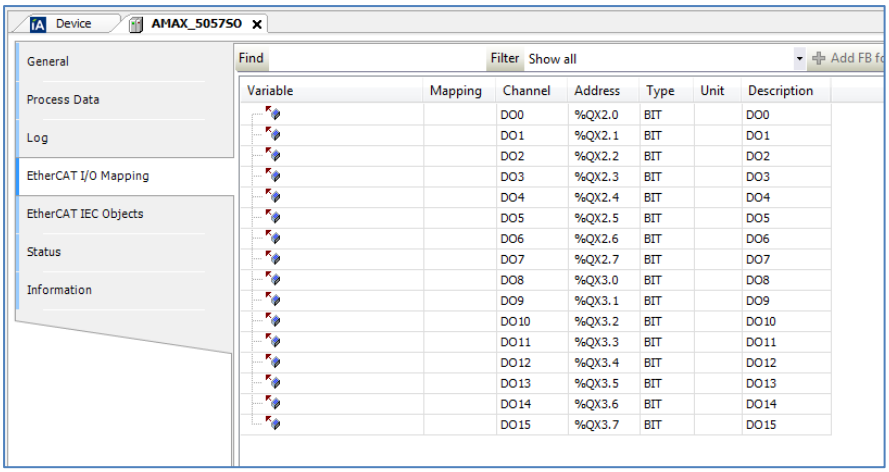


5.2.7. Digital Output Modules

In this section, we are going to introduce AMAX-5000 digital output modules, and take AMAX-5057SO as example. The **device editor** dialog window can be opened by double clicking the device name in the device tree.

In the **General tab**, user can easily identify the EtherCAT address has been assigned automatically. And all the information is from device description file. User can also unfold the **Distributed Clock** option to change the related setting. Usually user can directly go the **EtherCAT I/O Mapping** tab to map variables to the physical I/O.





Variable: Double click the column and select the proper variable for mapping.

Mapping: The mapping status of each variable.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 digital inputs. This will require either 8 Boolean addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

Q = Physical Output

X = Single bit

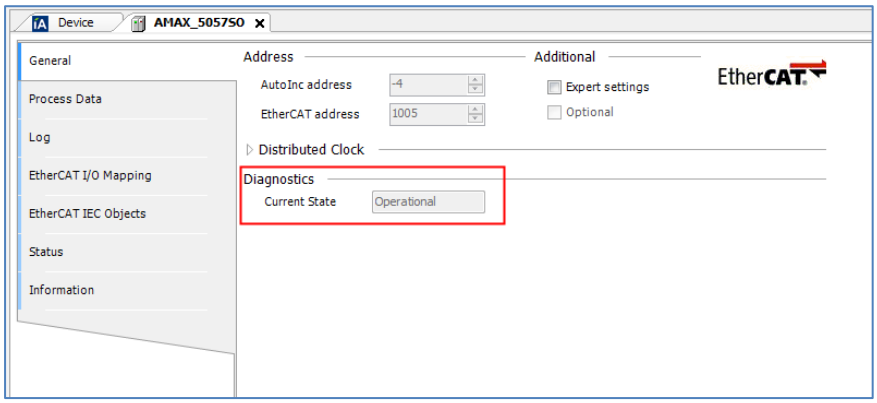
$(N1). $(N2) = Starting address. The first number means the starting byte; the second number means the starting bit.

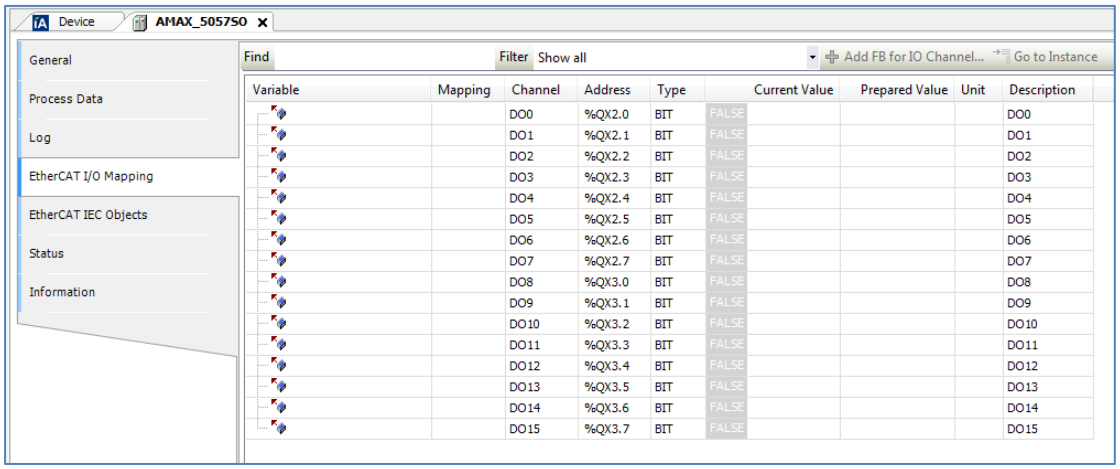
Type: The data type of each variable.

Unit: Reserved Column

Description: The description of each variable.

User can also **Login** to be on-line and check the status of modules.

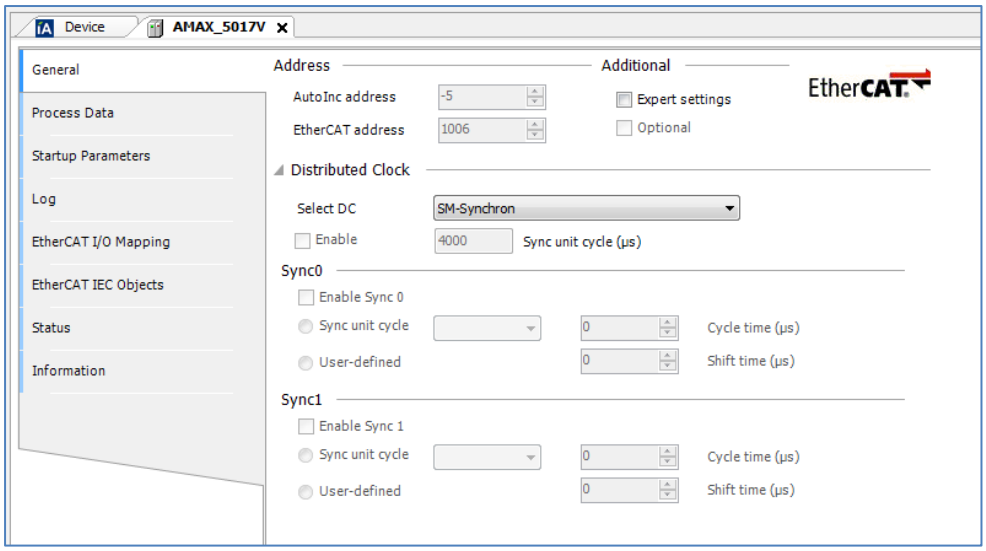

Device
AMAX_5057SO
General
Process Data
Log
EtherCAT I/O Mapping
EtherCAT IEC Objects
Status
Information
Address
Additional
AutoInc address
-4
EtherCAT address
1005
Expert settings
Optional
EtherCAT
Distributed Clock
Diagnostics
Current State
Operational

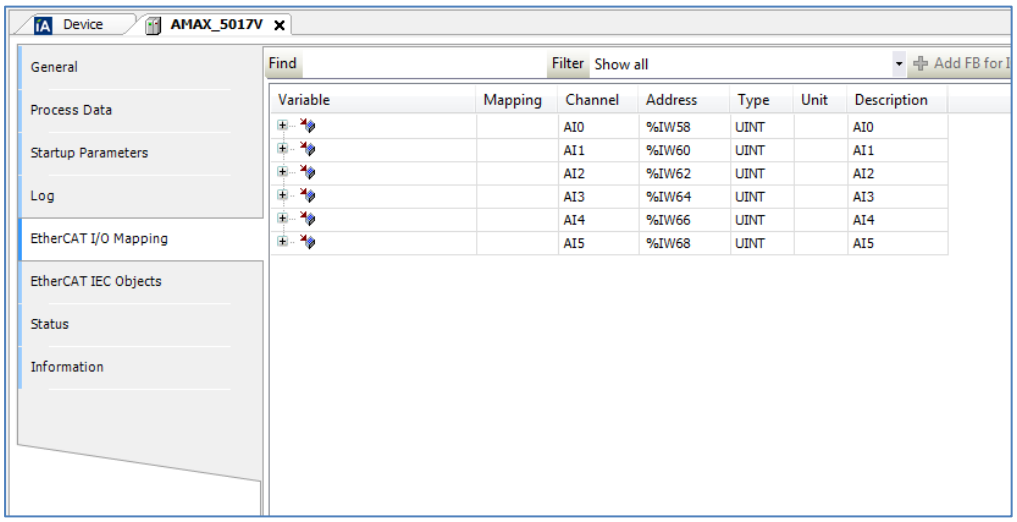

Device
AMAX_5057SO
General
Process Data
Log
EtherCAT I/O Mapping
EtherCAT IEC Objects
Status
Information
Find
Filter
Show all
Add FB for IO Channel...
Go to Instance
Variable
Mapping
Channel
Address
Type
Current Value
Prepared Value
Unit
Description
DO0 %QX2.0 BIT FALSE DO0
DO1 %QX2.1 BIT FALSE DO1
DO2 %QX2.2 BIT FALSE DO2
DO3 %QX2.3 BIT FALSE DO3
DO4 %QX2.4 BIT FALSE DO4
DO5 %QX2.5 BIT FALSE DO5
DO6 %QX2.6 BIT FALSE DO6
DO7 %QX2.7 BIT FALSE DO7
DO8 %QX3.0 BIT FALSE DO8
DO9 %QX3.1 BIT FALSE DO9
DO10 %QX3.2 BIT FALSE DO10
DO11 %QX3.3 BIT FALSE DO11
DO12 %QX3.4 BIT FALSE DO12
DO13 %QX3.5 BIT FALSE DO13
DO14 %QX3.6 BIT FALSE DO14
DO15 %QX3.7 BIT FALSE DO15

5.2.8. Analog Input Modules

In this section, we are going to introduce AMAX-5000 Analog input modules, and take AMAX-5017V as example. The **device editor** dialog window can be opened by double clicking the device name in the device tree.

In the **General tab**, user can easily identify the EtherCAT address has been assigned automatically. And all the information is from device description file. User can also unfold the **Distributed Clock** option to change the related setting. Usually user can directly go the **EtherCAT I/O Mapping** tab to map variables to the physical I/O.





Variable: Double click the column and select the proper variable for mapping.

Mapping: The mapping status of each variable.

Address: The starting physical address of the variables for this I/O group. The board shown below has 6 analog inputs. This will require 6 WORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

I = Physical Input

W = Word (16 Bits)

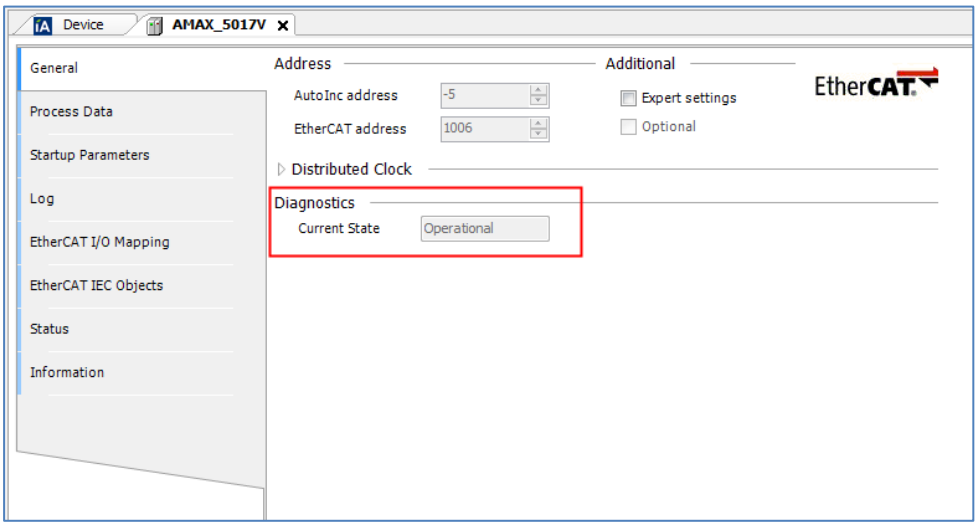
$(N) = Starting address.

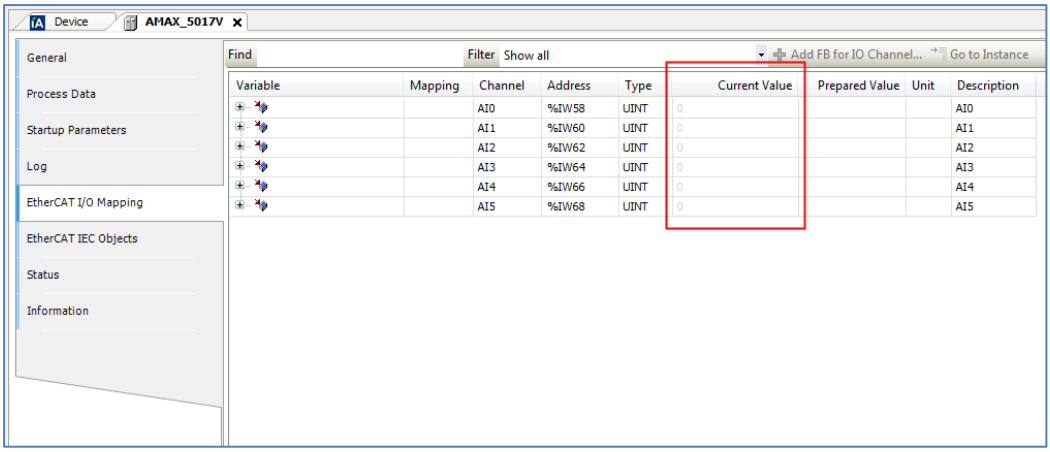
Type: The data type of each variable.

Unit: Reserved Column

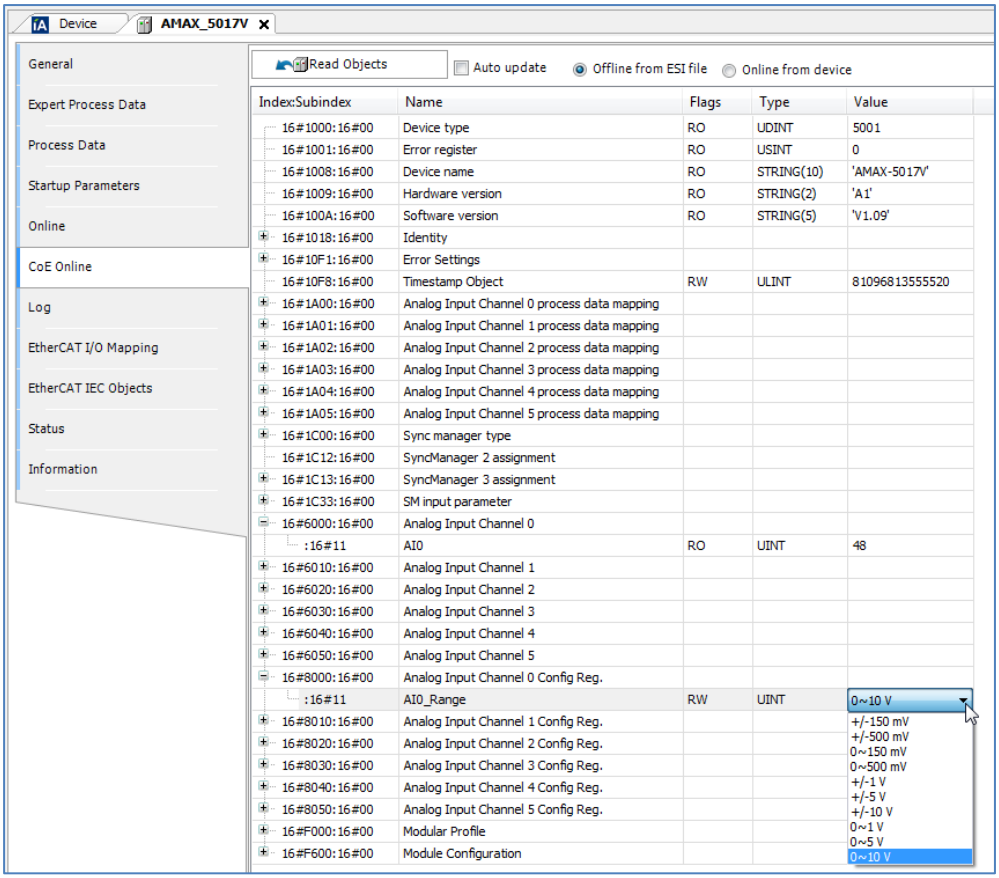
Description: The description of each variable.

User can also **Login** to be on-line and check the status of modules.





If user would like to change the working parameter of module, please select the **Expert settings** in the **General** tab, and click the **CoE Online** tab.

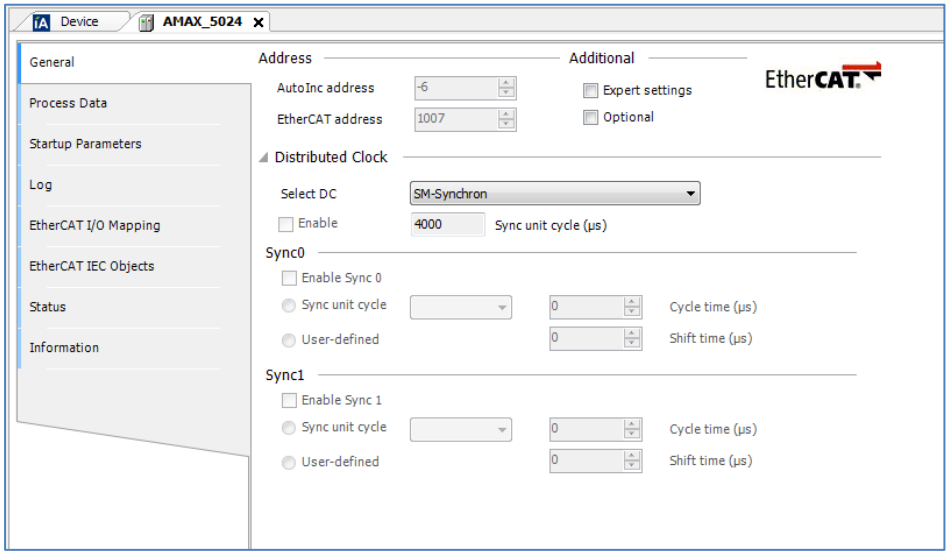


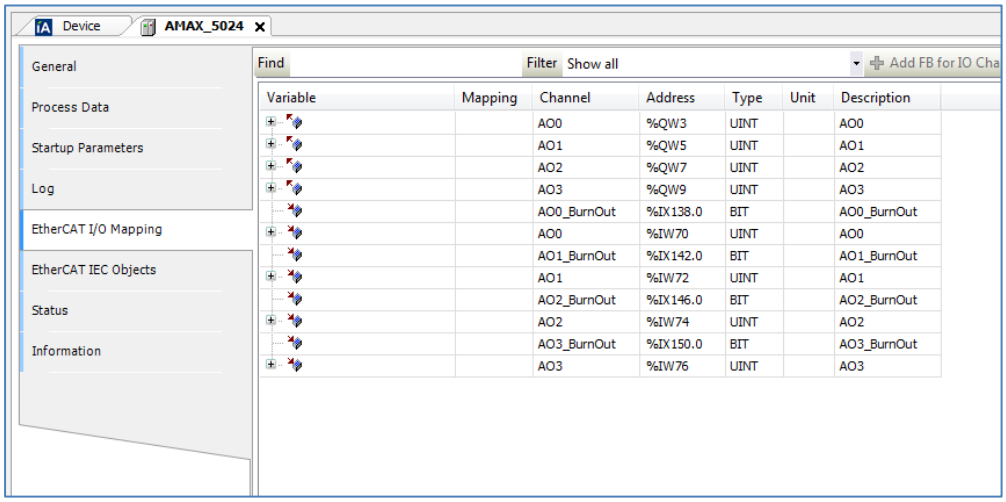
Select the target parameters from the table and double click the **Value** column. Select the target settings from the drop-down manual.

5.2.9. Analog Output Modules

In this section, we are going to introduce AMAX-5000 Analog output modules, and take AMAX-5024 as example. The **device editor** dialog window can be opened by double clicking the device name in the device tree.

In the **General tab**, user can easily identify the EtherCAT address has been assigned automatically. And all the information is from device description file. User can also unfold the **Distributed Clock** option to change the related setting. Usually user can directly go the **EtherCAT I/O Mapping** tab to map variables to the physical I/O.





Variable: Double click the column and select the proper variable for mapping.

Mapping: The mapping status of each variable.

Address: The starting physical address of the variables for this I/O group. The board shown below has 4 analog outputs. This will require 4 WORD addresses.

Note!

Meaning of address expression:

% = Direct Mapped variable

Q = Physical Output

W = Word (16 Bits)

$(N) = Starting address.

Type: The data type of each variable.

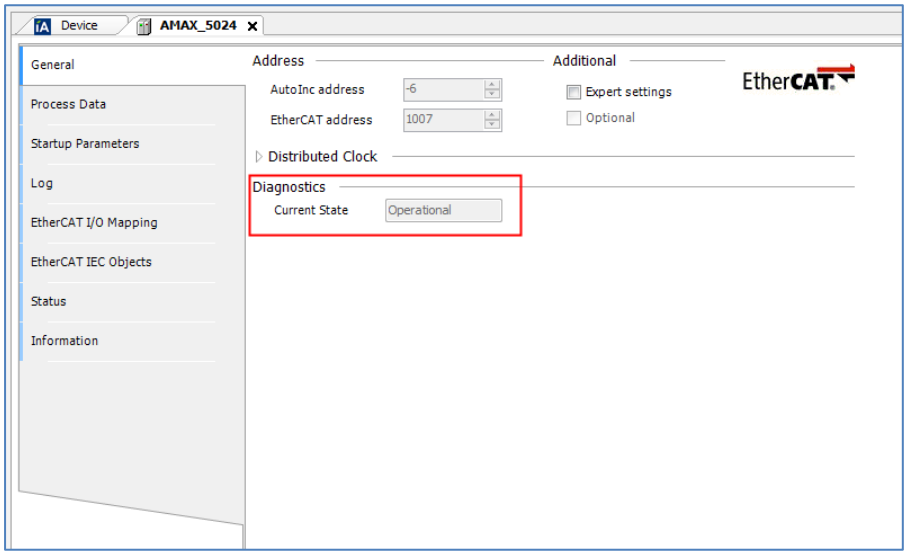
Unit: Reserved Column

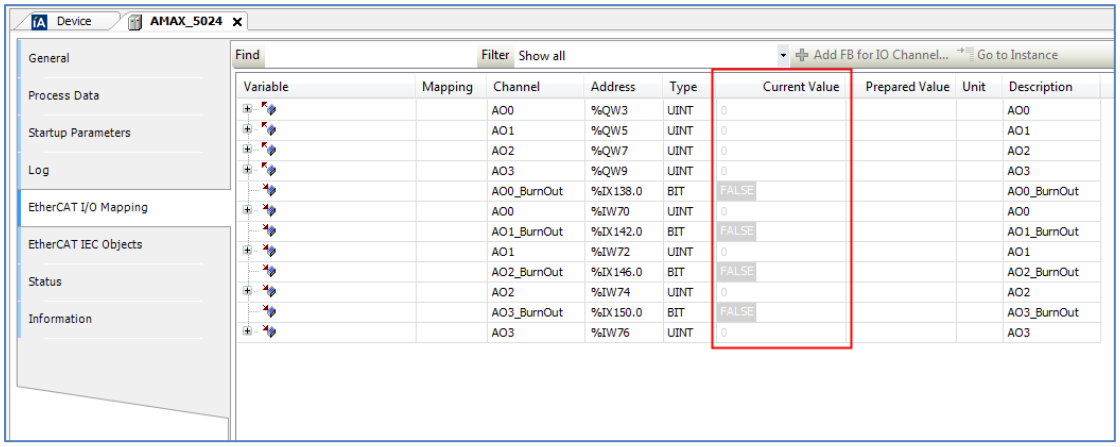
Description: The description of each variable.

Note!

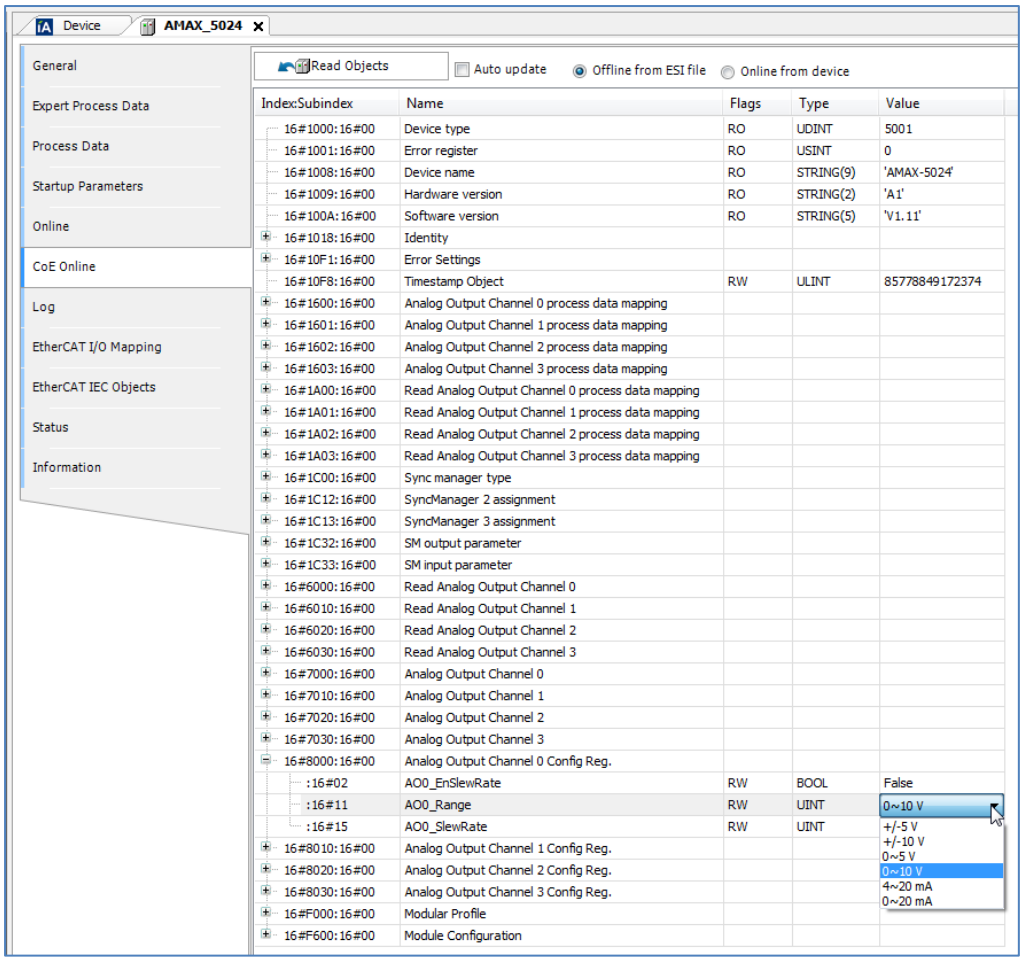
AMAX-5024 support AO read back function; please use the %IWxx to watch the real output of the module. For example, if users setup a slow slew rate of AO, user can use this variable to check the current real output value in each cycle.

User can also **Login** to be on-line and check the status of modules.





If user would like to change the working parameter of module, please select the **Expert settings** in the **General** tab, and click the **CoE Online** tab.

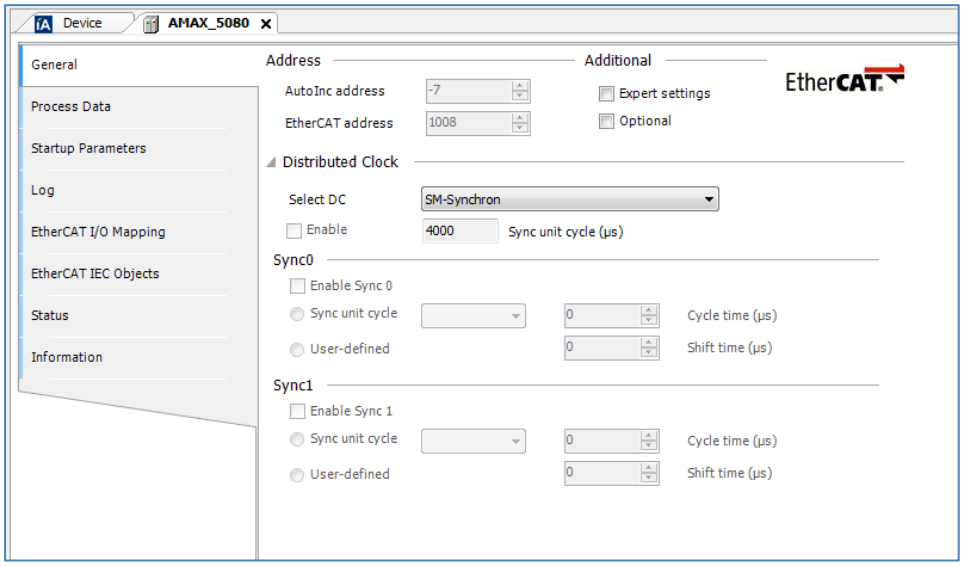


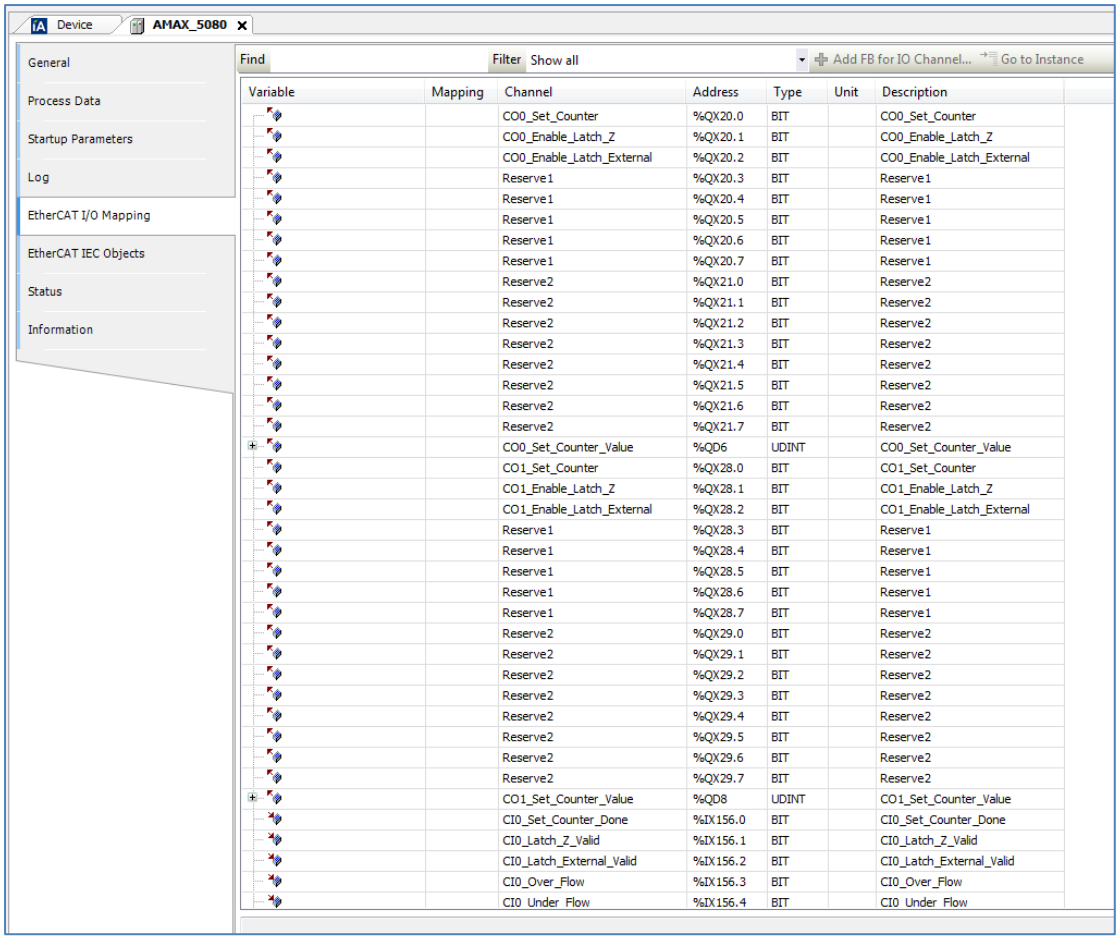
Select the target parameters from the table and double click the **Value** column. Select the target settings from the drop-down manual.

5.2.10. Counter Modules

In this section, we are going to introduce AMAX-5000 Counter modules, and take AMAX-5080 as example. The **device editor** dialog window can be opened by double clicking the device name in the device tree.

In the **General tab**, user can easily identify the EtherCAT address has been assigned automatically. And all the information is from device description file. User can also unfold the **Distributed Clock** option to change the related setting. Usually user can directly go the **EtherCAT I/O Mapping** tab to map variables to the physical I/O.





Variable	Mapping	Channel	Address	Type	Unit	Description
		CO0_Set_Counter	%QX20.0	BIT		CO0_Set_Counter
		CO0_Enable_Latch_Z	%QX20.1	BIT		CO0_Enable_Latch_Z
		CO0_Enable_Latch_External	%QX20.2	BIT		CO0_Enable_Latch_External
		Reserve1	%QX20.3	BIT		Reserve1
		Reserve1	%QX20.4	BIT		Reserve1
		Reserve1	%QX20.5	BIT		Reserve1
		Reserve1	%QX20.6	BIT		Reserve1
		Reserve1	%QX20.7	BIT		Reserve1
		Reserve2	%QX21.0	BIT		Reserve2
		Reserve2	%QX21.1	BIT		Reserve2
		Reserve2	%QX21.2	BIT		Reserve2
		Reserve2	%QX21.3	BIT		Reserve2
		Reserve2	%QX21.4	BIT		Reserve2
		Reserve2	%QX21.5	BIT		Reserve2
		Reserve2	%QX21.6	BIT		Reserve2
		Reserve2	%QX21.7	BIT		Reserve2
		CO0_Set_Counter_Value	%QD6	UDINT		CO0_Set_Counter_Value
		CO1_Set_Counter	%QX28.0	BIT		CO1_Set_Counter
		CO1_Enable_Latch_Z	%QX28.1	BIT		CO1_Enable_Latch_Z
		CO1_Enable_Latch_External	%QX28.2	BIT		CO1_Enable_Latch_External
		Reserve1	%QX28.3	BIT		Reserve1
		Reserve1	%QX28.4	BIT		Reserve1
		Reserve1	%QX28.5	BIT		Reserve1
		Reserve1	%QX28.6	BIT		Reserve1
		Reserve1	%QX28.7	BIT		Reserve1
		Reserve2	%QX29.0	BIT		Reserve2
		Reserve2	%QX29.1	BIT		Reserve2
		Reserve2	%QX29.2	BIT		Reserve2
		Reserve2	%QX29.3	BIT		Reserve2
		Reserve2	%QX29.4	BIT		Reserve2
		Reserve2	%QX29.5	BIT		Reserve2
		Reserve2	%QX29.6	BIT		Reserve2
		Reserve2	%QX29.7	BIT		Reserve2
		CO1_Set_Counter_Value	%QD8	UDINT		CO1_Set_Counter_Value
		CI0_Set_Counter_Done	%IX156.0	BIT		CI0_Set_Counter_Done
		CI0_Latch_Z_Valid	%IX156.1	BIT		CI0_Latch_Z_Valid
		CI0_Latch_External_Valid	%IX156.2	BIT		CI0_Latch_External_Valid
		CI0_Over_Flow	%IX156.3	BIT		CI0_Over_Flow
		CI0_Under_Flow	%IX156.4	BIT		CI0_Under_Flow

Variable: Double click the column and select the proper variable for mapping.

Mapping: The mapping status of each variable.

Address: The starting physical address of the variables for this I/O group. The board shown below has 4 analog outputs. This will require 4 WORD addresses.

Note!

Meaning of address expression:

% = Direct Mapped variable

Q = Physical Output

I = Physical Input

X = Single bit

W = Word (16 Bits)

$(N) = Starting address.

Type: The data type of each variable.

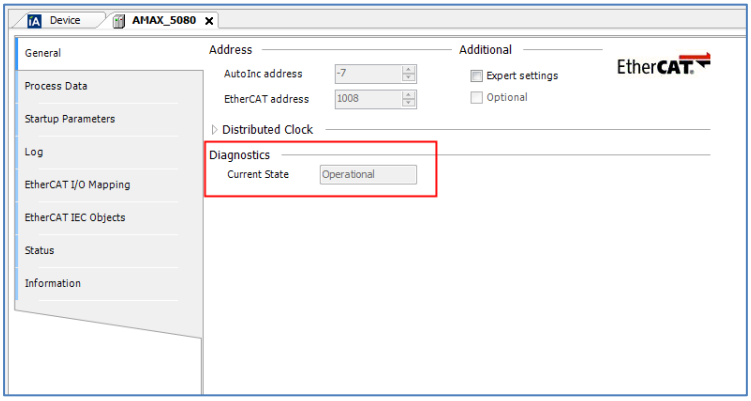
Unit: Reserved Column

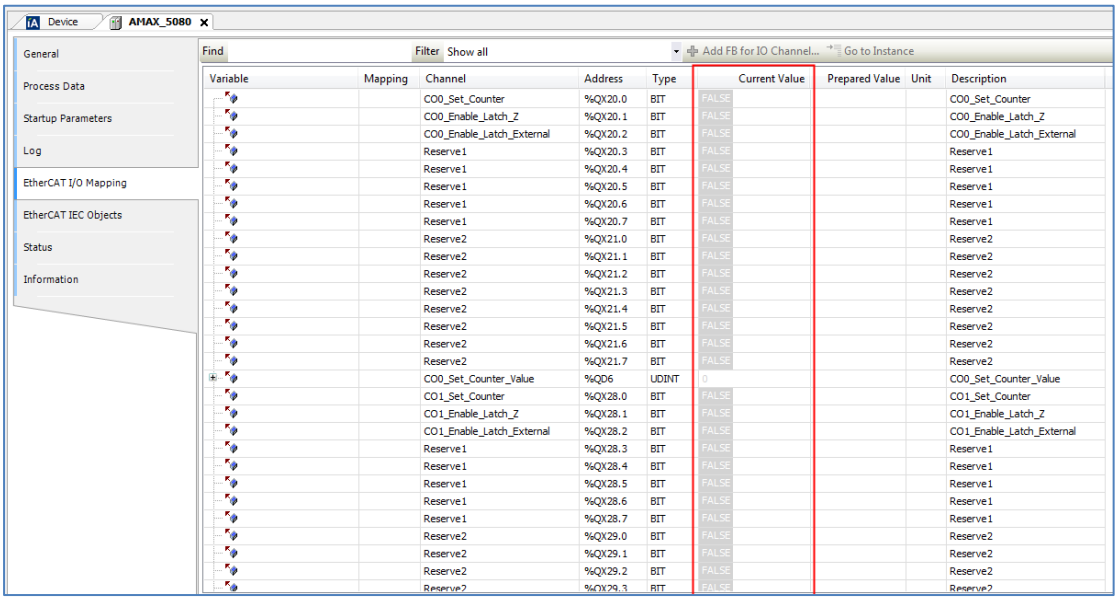
Description: The description of each variable.

Note!

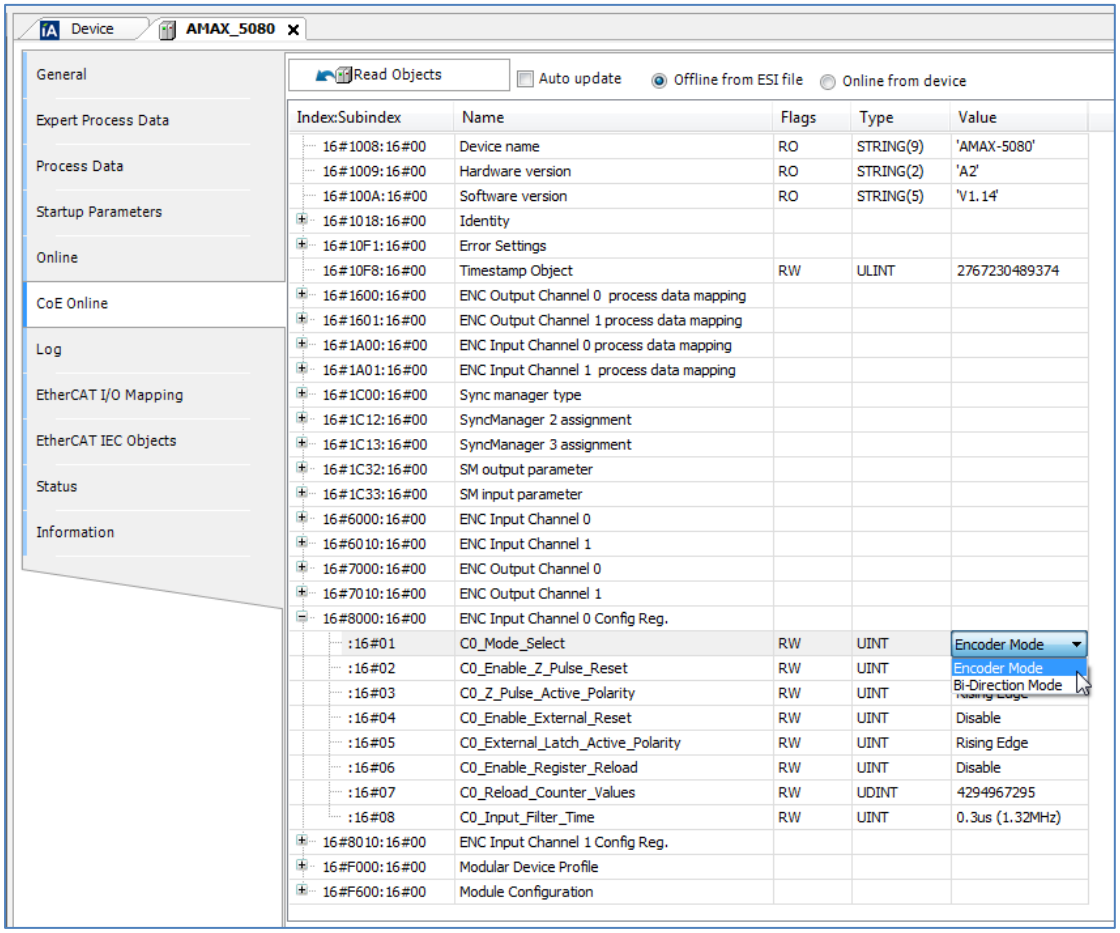
AMAX-5080 support many kinds of operation modes and parameter, please refer to the hardware user's manual for detail.

User can also **Login** to be on-line and check the status of modules.





If user would like to change the working parameter of module, please select the **Expert settings** in the **General** tab, and click the **CoE Online** tab.



Select the target parameters from the table and double click the **Value** column. Select the target settings from the drop-down manual.

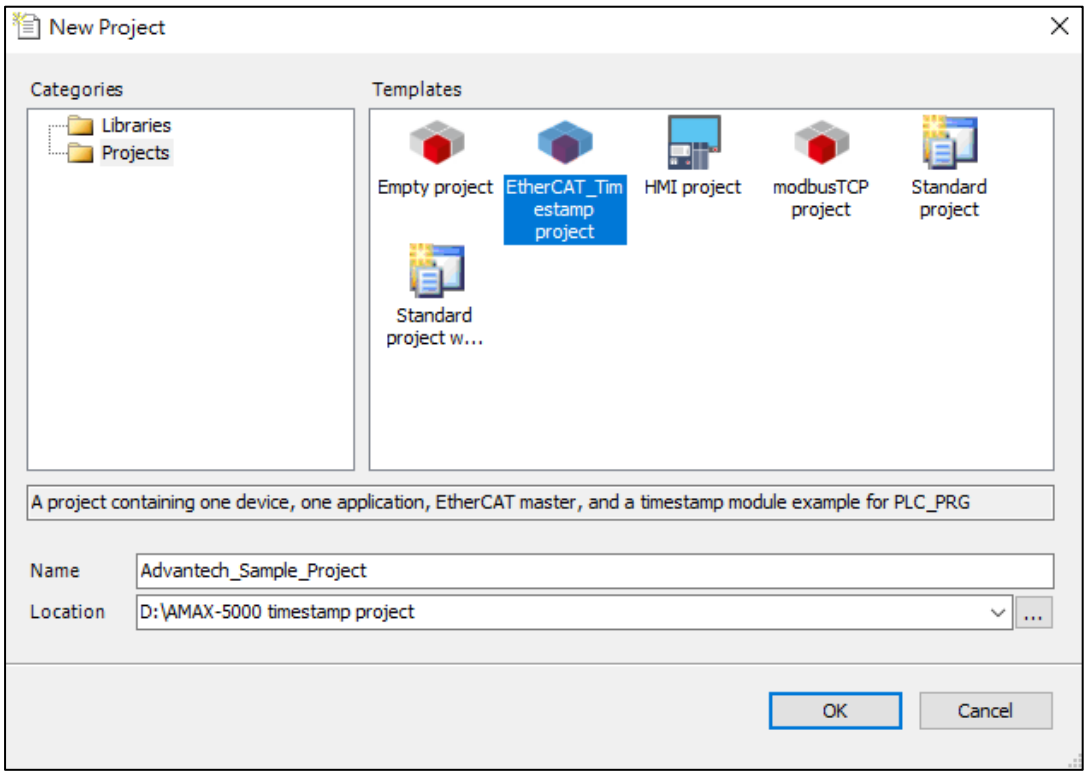
5.2.11. Timestamp Modules

In this section, we'll introduce the CoDeSys function block and sample code for AMAX-5051T and AMAX-5056T timestamp modules.

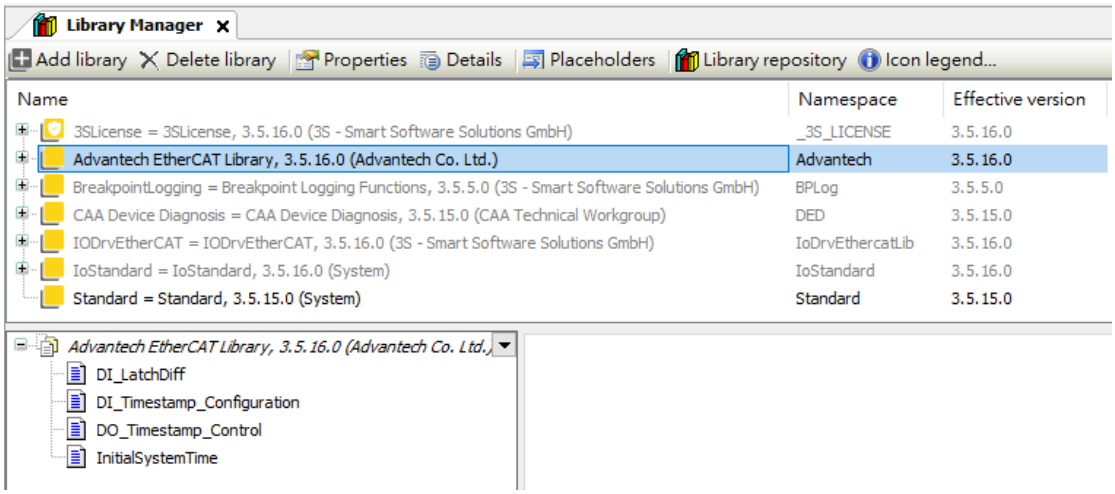
(Note: Please refer to AMAX-5000 user's manual chapter7 for more detailed product features.)

5.2.11.1 EtherCAT Timestamp Templates

By the installation of the Advantech add-on package, you can find the “EtherCAT_Timestamp project” in the template area.



The new project created by the template contains four function blocks and a sample program; this document will introduce the definition of the function blocks.

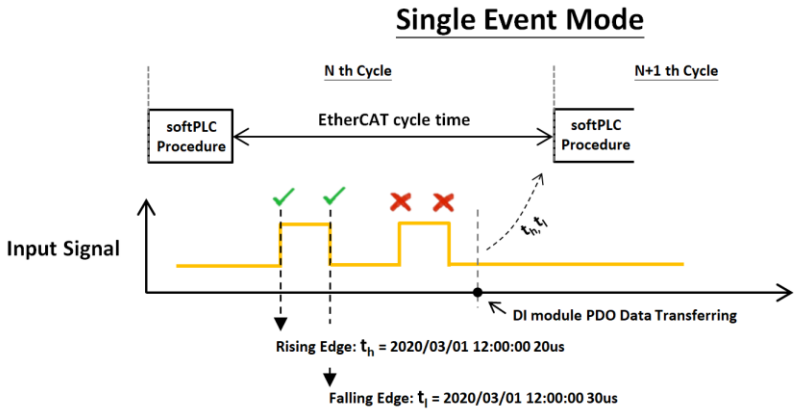


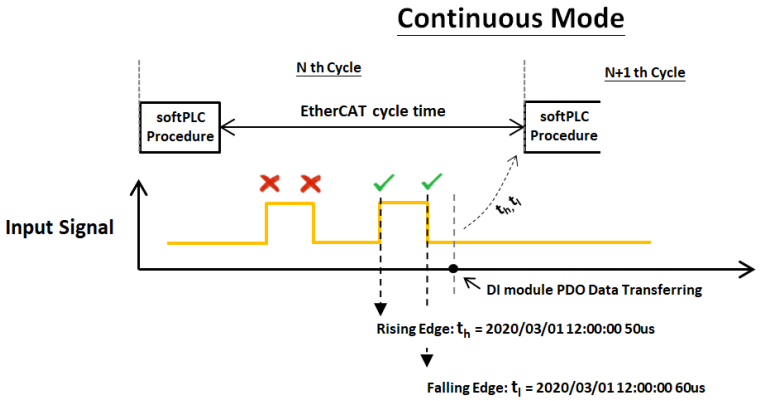
5.2.11.2 AMAX-5051T Digital Input with Timestamp

In order to latch the input signal, the digital input module contains a set of parameters to record the precise timestamp for each rising-edge (th) and falling-edge (tl). One thing to be noticed, is that there is only one pair of timestamp (for th and tl) can be stored in the module, so the user should select the latching mode: Single Event or Continuous (default).

The Single Event mode only latches the first rising-edge and falling-edge timestamp and ignores any state change afterward. The Continuous mode will continuously update the latest timestamp of state changes.

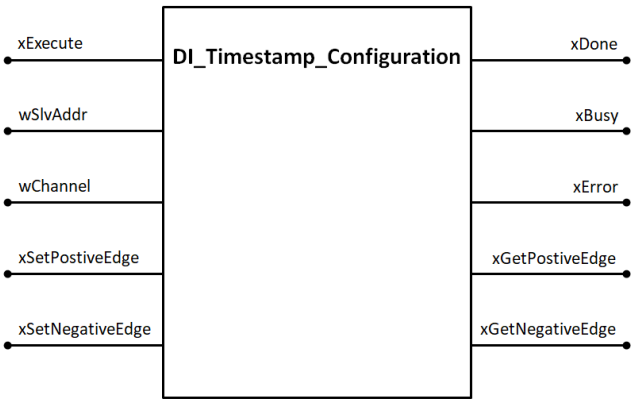
Each rising-edge (th) and falling-edge (tl) can be set to Single Event mode or Continuous mode independently.





Function Block – DI_Timestamp_Configuration

This function block provides an easy way to configure the mode of active edge.



Input	Data Type	Definition

xExecute	BOOL	Execute the function block
wSlvAddr	WORD	The target slave's address
wChannel	WORD	The target slave channel
xSetPositiveEdge	BOOL	Select mode for positive edge timestamping: True: Single Event Mode False: Continuous Mode
xSetNegativeEdge	BOOL	Select mode for negative edge timestamping: True: Single Event Mode False: Continuous Mode

Output	Data Type	Definition
xDone	BOOL	The function block's task is Done
xBusy	BOOL	The function block's task is Busy
xError	BOOL	The function block's task is Error
xGetPostiveEdge	BOOL	Positive edge timestamping mode
xGetNegativeEdge	BOOL	Negative edge timestamping mode

Function Block – DI_LatchDiff

The DI_LatchDiff function block provides two functions:

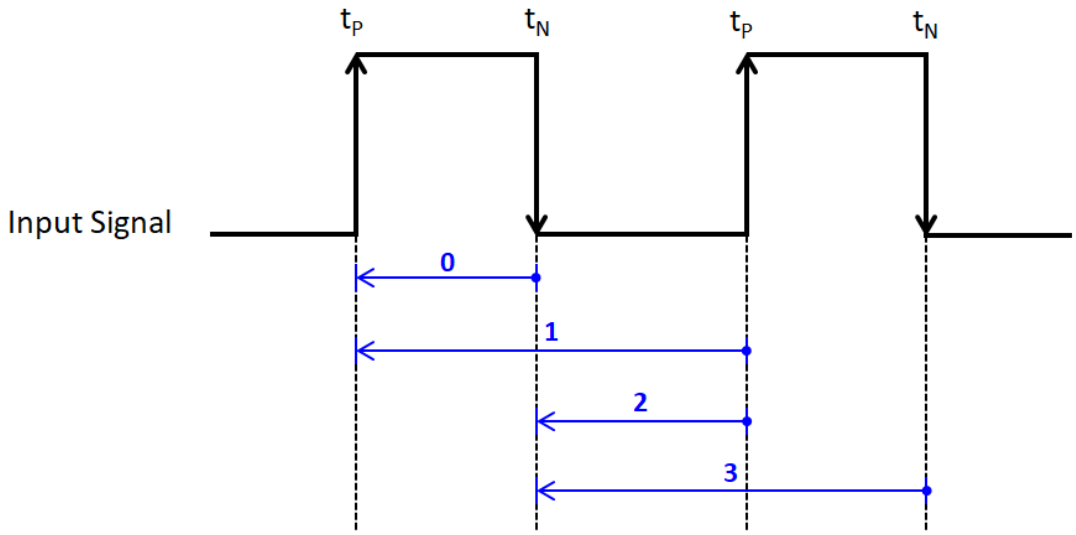
1. Translate the data time base, and divide into two parts. (seconds, and fractional seconds)
2. Calculate the time difference between two input signal edges.

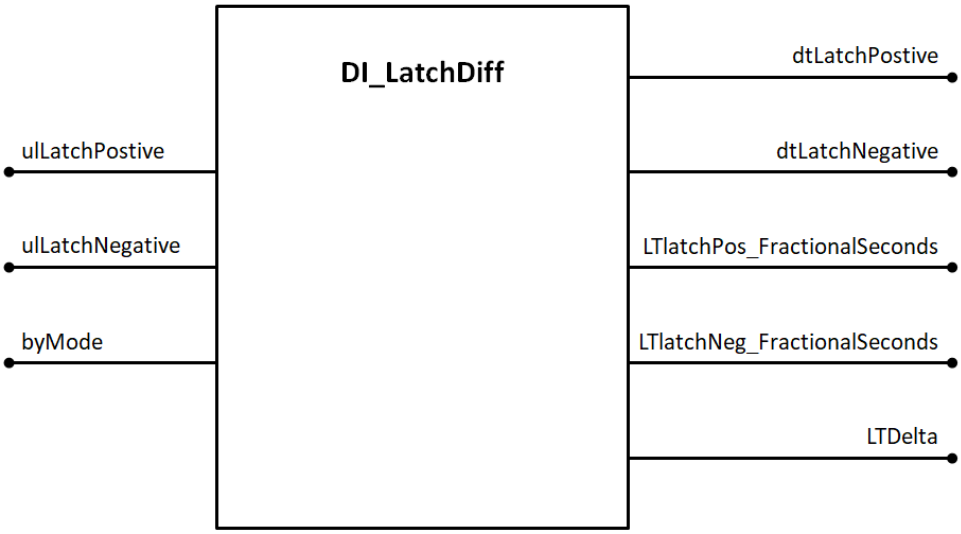
The Time System Translation

The time system for EtherCAT distribute clocks is start from **2000-01-01, 00:00:00**, 64-bits range with 1ns resolution. And the time system in DT format in CoDeSys is start from **1970-01-01, 00:00:00**. So some translation must be done before the use.

Calculation of the time difference between two edges

Object detection is one of the most common digital input applications in automation industry, the precise time difference between two input signal edges can be critical for some applications. The DI_LatchDiff function block provides a method to calculate 4 different forms of the time difference like figure below:



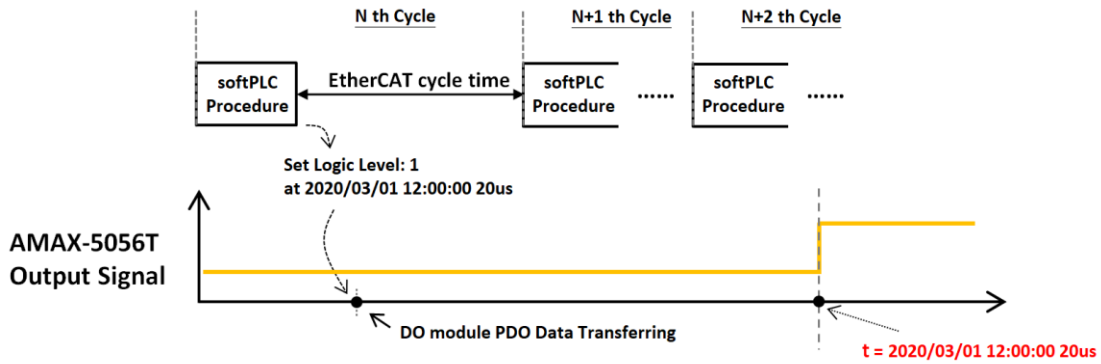


Input	Data Type	Definition
ulLatchPostive	ULINT	Mapping with the variable of AMAX-5051T LatchPos0 or LatchPos1
ulLatchNegative	ULINT	Mapping with the variable of AMAX-5051T LatchNeg0 or LatchNeg1
byMode	BYTE	Mode Selection: 0: Pos_Neg = The time difference between previous ulLatchPostive and current ulLatchNegative. 1: Pos_Pos = The time difference between previous ulLatchPostive and current ulLatchPostive. 2: Neg_Pos = The time difference between previous ulLatchNegative and current ulLatchPostive. 3: Neg_Neg = The time difference between previous ulLatchNegative and current ulLatchNegative.

Output	Data Type	Definition
dtLatchPostive	DT	Transform the **ulLatchPostive** data format to date time (in the unit of sec) e.g.
dtLatchNegative	DT	Transform the **ulLatchNegative** data format to date time (in the unit of sec)
LTlatchPos_FractionalSeconds	LTIME	The fractional seconds of **ulLatchPostive** (in the unit of ns)
LTlatchNeg_FractionalSeconds	LTIME	The fractional seconds of **ulLatchNegative** (in the unit of ns)
LTDelta	LTIME	The time difference between edges (in the unit of ns)

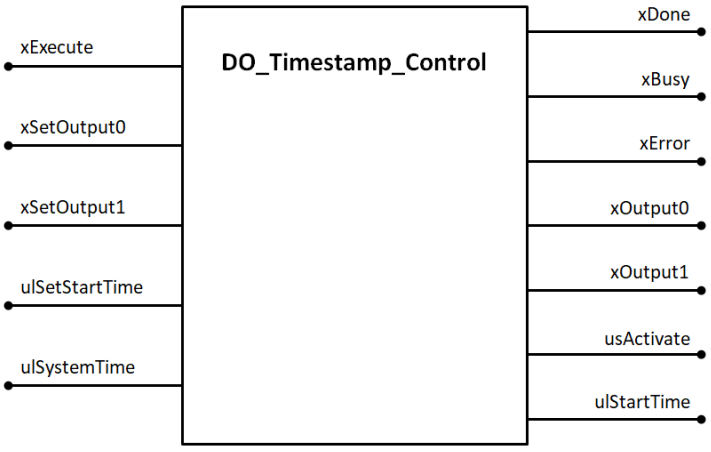
5.2.11.3 AMAX-5056T Digital Output with Timestamp

By setting Start Time and Activation to the time-stamping digital output module, the preset logic level will be activated at any specific time of the cycle as figure below.



Function Block – DO_Timestamp_Control

This function block provides a way to set the trigger time for the DO0 and DO1. Please note that there is only one timestamp register in the module, which means two channels of DO are sharing one timestamp, so the triggering signal can only be sent one at a time.



Input	Data Type	Definition
xExecute	BOOL	Execute the function block
xSetOutput0	BOOL	Setting the logic level of DO0 at start time
xSetOutput1	BOOL	Setting the logic level of DO1 at start time
ulSetStartTime	ULINT	Setting the trigger time of both DO0 and DO1
ulSystemTime	ULINT	Current Time

Output	Data Type	Definition
xDone	BOOL	The function block's task is Done
xBusy	BOOL	The function block's task is Busy
xError	BOOL	The function block's task is Error
xOutput0	BOOL	The output level of DO0
xOutput1	BOOL	The output level of DO1
usActivate	USINT	The activation of the slave PDO
ulStartTime	ULINT	The output start time

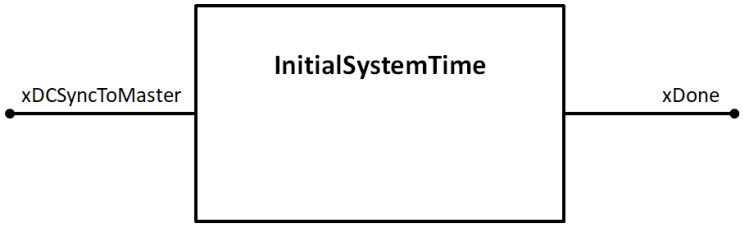
5.2.11.4 EtherCAT Distributed Clocks Synchronization

Function Block – InitialSystemTime

There are two ways to synchronize EtherCAT distribution clock:

1. Synchronize with master system clock.
2. Synchronize with first slave module.

This InitialSystemTime function block provides a way to set the method of distribution clock Synchronization.



Input	Data Type	Definition
xDCSyncToMaster	BOOL	TRUE: Synchronize all slaves time with the master's system time. FALUE: Synchronize all slaves time with the first slave's time.

Output	Data Type	Definition
xDone	BOOL	The function block’s task is Done

Chapter 6

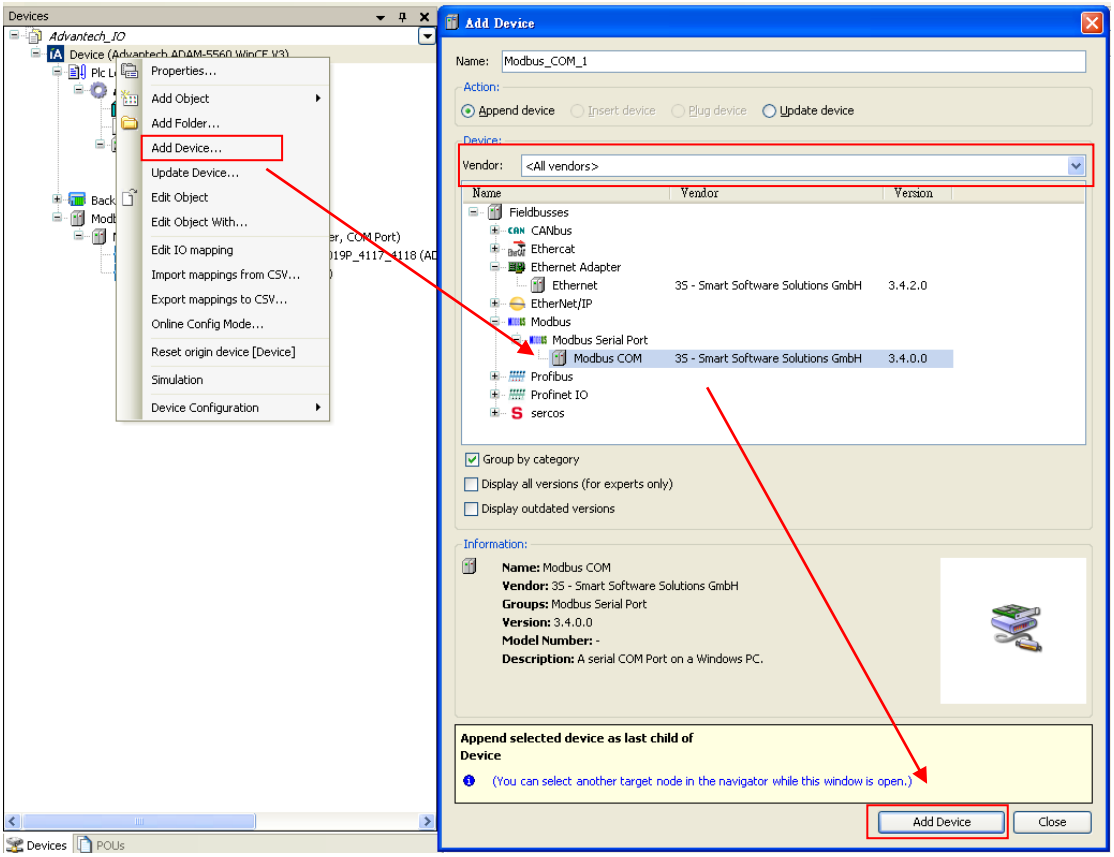
6. Advantech Fieldbus Modules

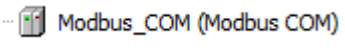
Advantech ADAM series distributed data acquisition and control systems are the ideal tools for creating multi-drop Fieldbus networks. The module design allows users to create application-specific configurations with ease. System communicates with their controlling host using either serial COM port or communication protocols.

6.1. Modbus

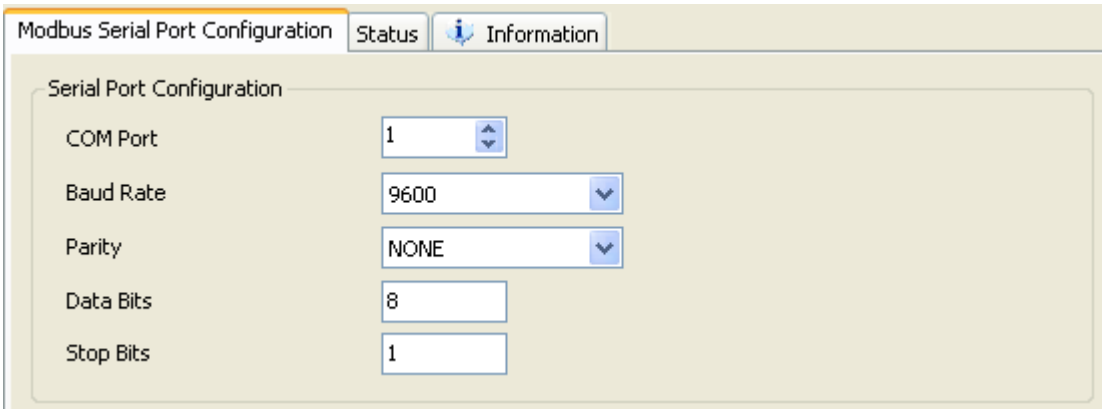
6.1.1. Modbus RTU Client

In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog. Type the device name in Name container. Choose **Modbus COM** in the **Modbus** option and click **Add Device** to proceed and then press Close to close the device dialog.



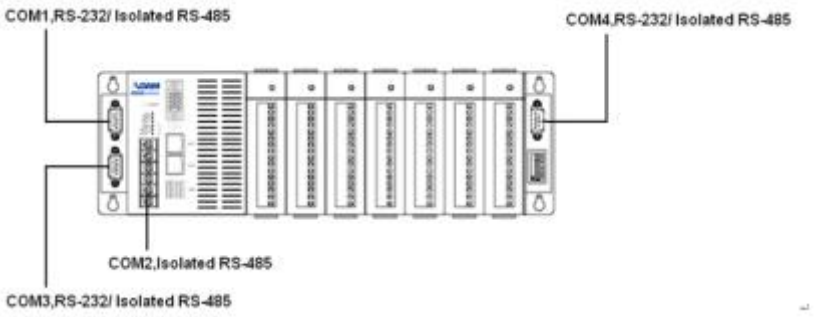
Now, you'll see Modbus COM 

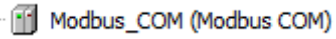
in the device tree. Double-click the Modbus COM icon to set configuration.

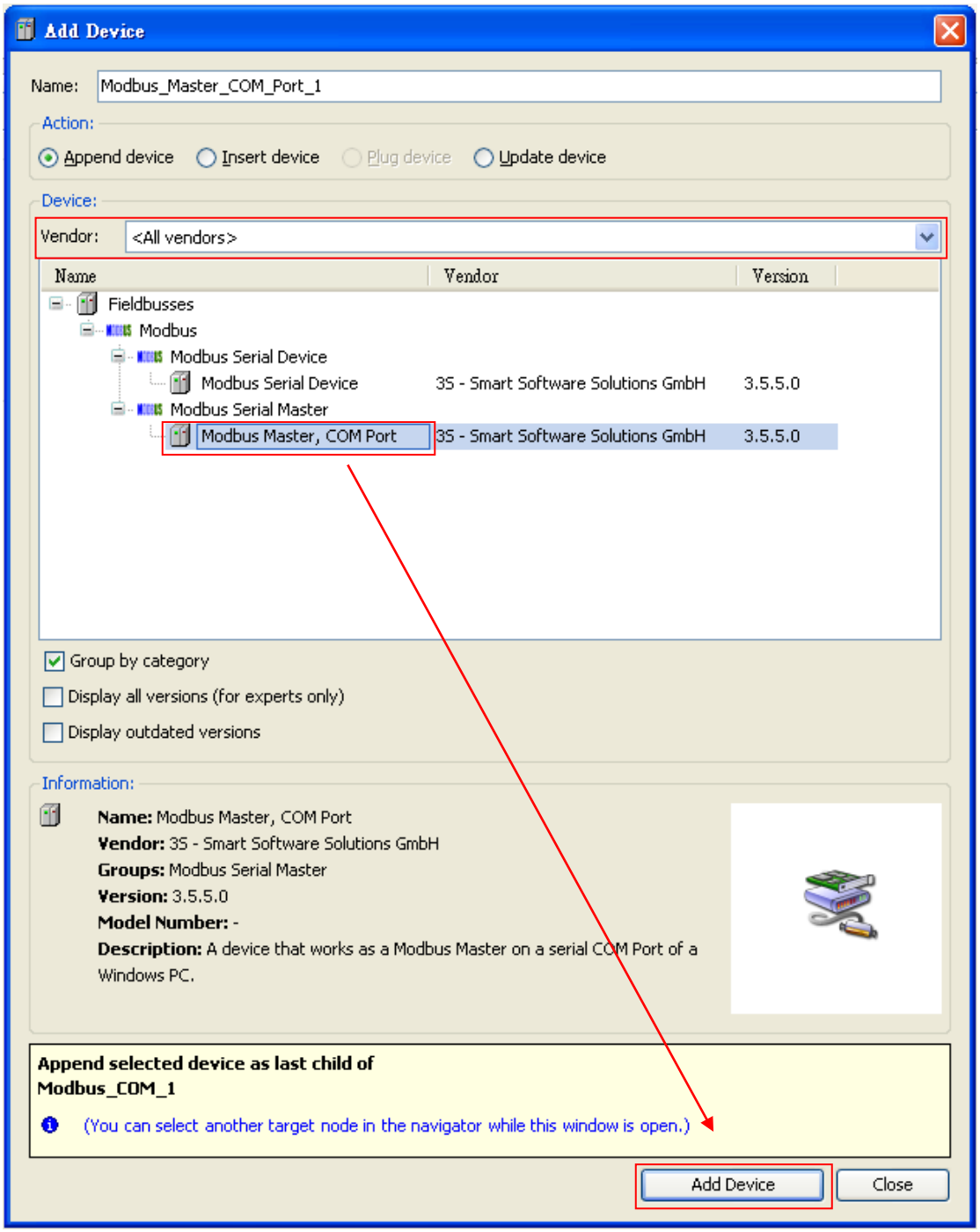


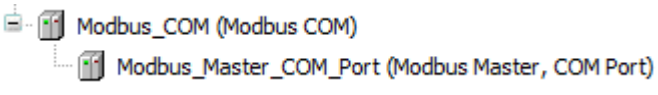
Note!

The location of COM ports is shown below. Follow the figure to set COM Port index.



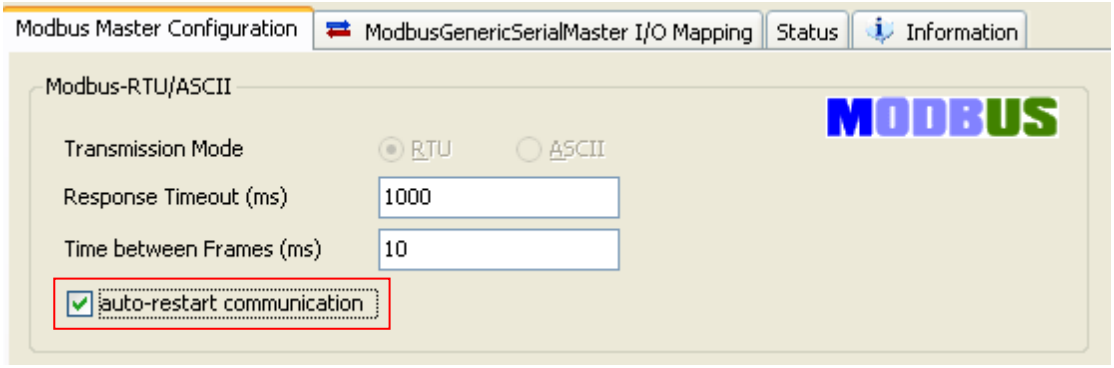
Right click on Modbus COM  in the device tree and click **Add Device.** Type the device name in Name container. Choose **Modbus Master, COM Port** in the **Modbus** option. Click **Add Device** to proceed and then press **Close** to close the device dialog.



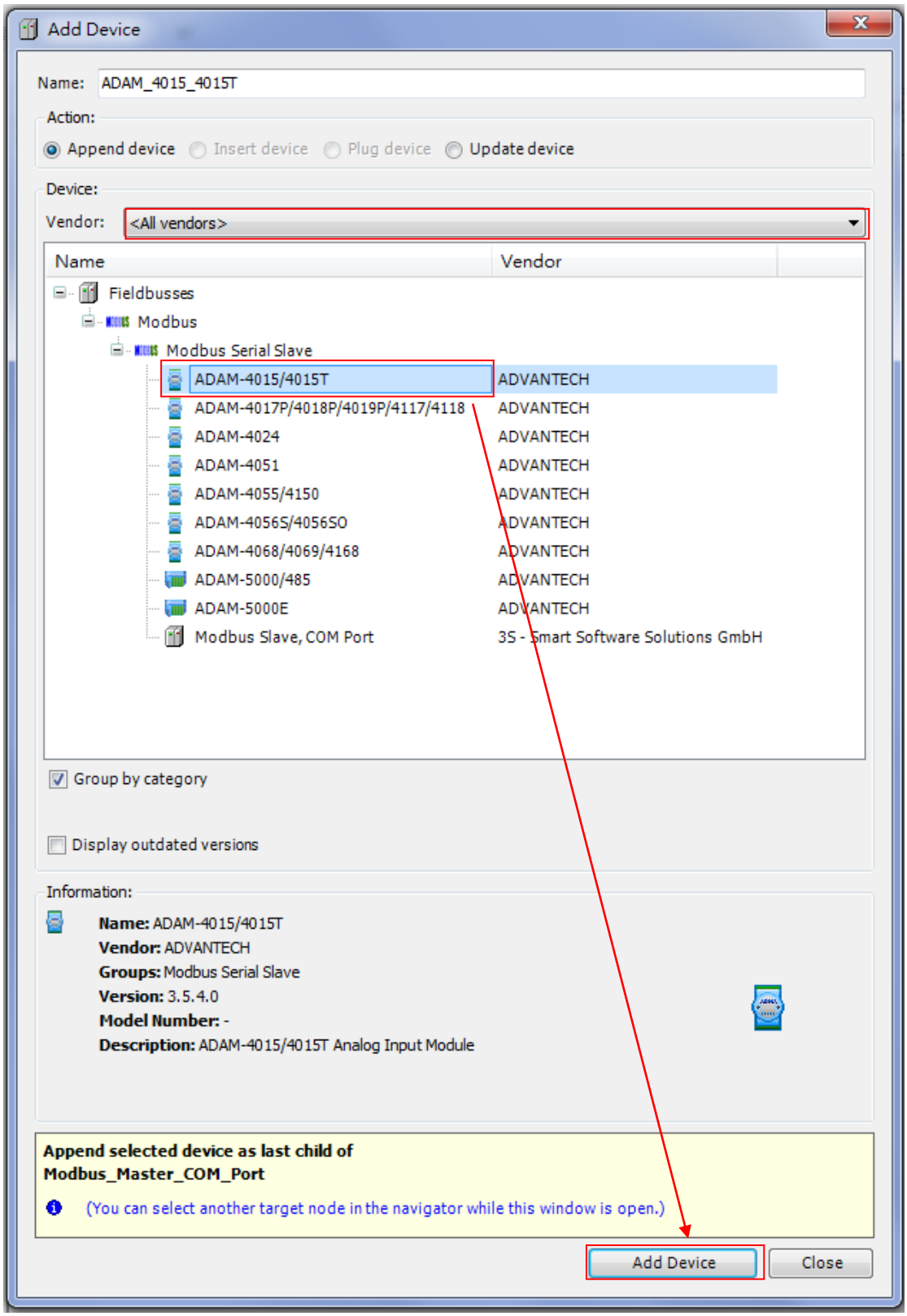
Now, you'll see Modbus master 

in the device tree. Double-click Modbus Master COM Port icon to set master configuration. For

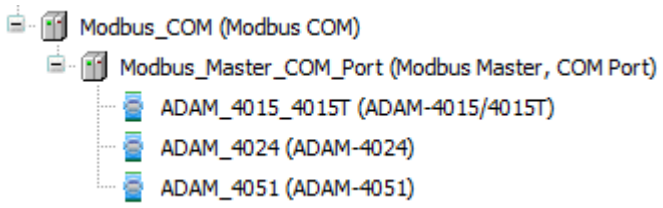
further convenience, recommend you to check **auto-restart communication**.



Right click on Modbus master in the device tree and click **Add Device** in context menu. It will open the **Add Device dialog**, where you choose one available device for the current connection. Type the device name in Name container. Click **Add Device** to proceed and then press **Close** to close the device dialog.



Now, you will see Advantech ADAM modules on device list.



Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. The editor is subdivided in the following tabs and the details will be described below:

Modbus Slave Configuration:

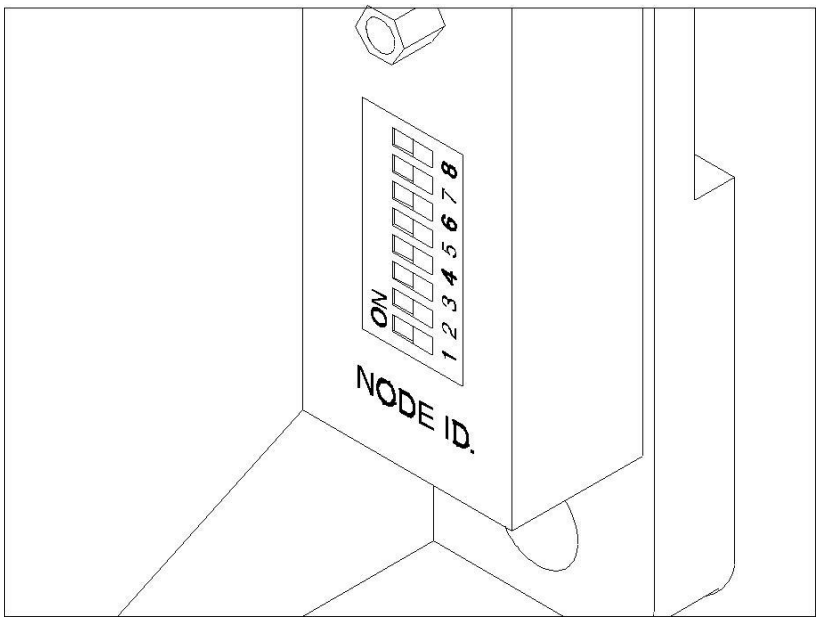
The following parameters deal with Modbus slave settings:

Slave Address: This number may range between 1 and 247; it serves to identify the address of a serial Modbus device.

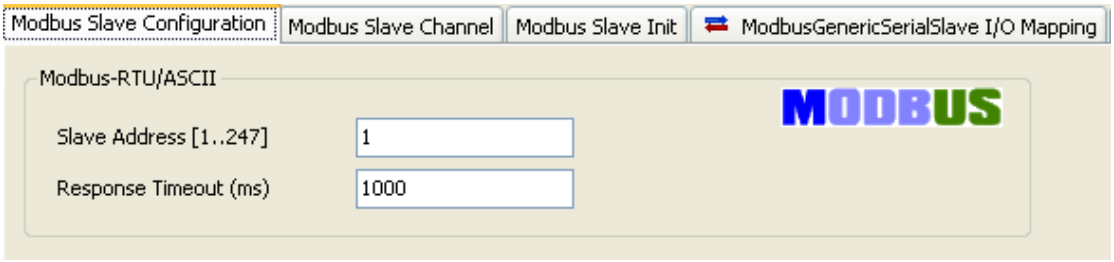
Note!

For ADAM-4000 series, move hardware switch to Initial mode and use **Advantech Adam/Apax Utility** to set slave address.

For ADAM-5000 series, use the 8-pin DIP switch to set slave address. Valid settings range from 0 to 127 where ON in any of the 8 DIP switch positions equates to a binary 1, and OFF equates to a binary 0. For initial setting, set address as 0 and baud rate setting will be fixed to 9600 bps. It is recommended to setting the range from 1 to 127.



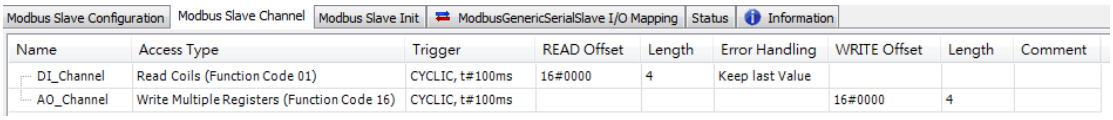
Response Timeout: Time interval for the master to wait for the response from the slave. This is especially configured for this slave node and overwrites the general response timeout setting of the respective master.



Modbus Slave Channel

This page is used to set up slave channels.

You can revise the default value by double-clicking the table. The following parameters deal with slave channel settings:



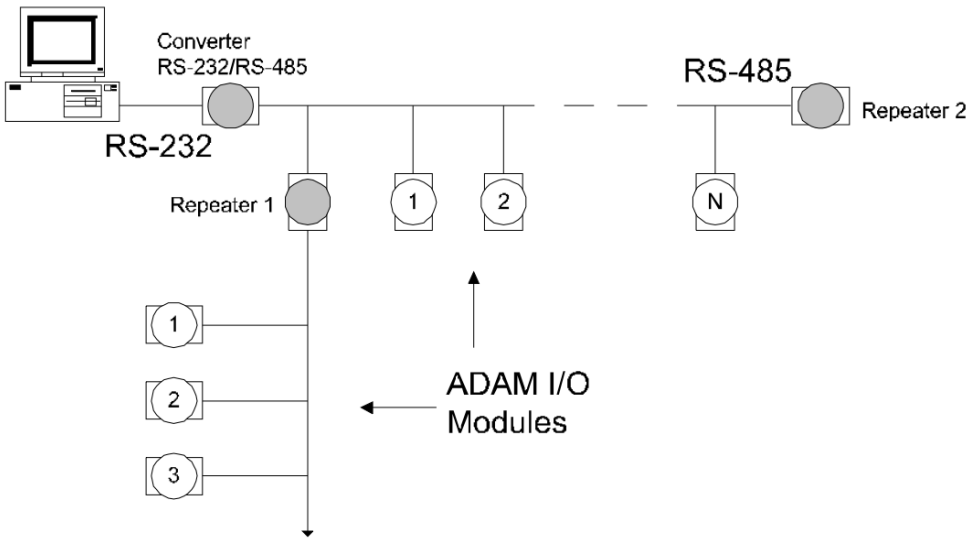
Column Name	Description
Name	The channel name
Access Type	Read Coil Status (Function Code 01) Read Holding Register (Function Code 03) Force Multiple Coils (Function Code 15) Preset Multiple Registers (Function Code 16)
Trigger	CYCLIC: The request occurs periodically. RISING_EDGE: The request occurs as a reaction to a rising edge of the Boolean trigger variables.
Cycle Time	If set Trigger as CYCLIC, it represents poll interval (in milliseconds) ***Note!*** **The poll interval should be the same as or a multiple of the cycle time of the application**

READ Offset	Start address where reading should start (Range 0-65535)
WRITE Offset	Start address where writing should start (Range 0-65535)
Length	Number of registers to be read/written (for word access) or number of discrete inputs to be read/written (for bit access)

We have already introduced **Modbus Slave Configuration** and **Modbus Slave Channel.**
In the following section, we will introduce **Modbus Slave I/O Mapping** for ADAM-4000 Series and ADAM-5000 Series according to their Modbus function code.

6.1.1.1. ADAM-4000 Series

The ADAM-4000 series is a set of intelligent sensor-to-computer interface modules containing built-in microprocessor. They are remotely controlled through a simple set of commands issued in ASCII format and transmitted in RS-485 protocol.
The figure below shows the brief overview of the ADAM-4000 system architecture.



Advantech provides 15 types of ADAM-4000 modules for various applications so far.
Following table is ADAM-4000 series support list.

Name	Specification
ADAM-4015	6-ch RTD Input Module

ADAM-4015T	6-ch Thermocouple Input Module
ADAM-4017+	8-ch Analog Input Module
ADAM-4018+	8-ch Analog Input Module
ADAM-4019+	8-ch Analog Input Module
ADAM-4117	8-ch Analog Input Module
ADAM-4118	8-ch Thermocouple Input Module
ADAM-4024	4-ch Analog Output Module
ADAM-4051	16-ch Digital Input Module
ADAM-4055	8-ch Digital Input and 8-ch Digital Output Module
ADAM-4150	8-ch Digital Input and 8-ch Digital Output Module
ADAM-4056S (SO)	12-ch Digital Output Module
ADAM-4068	8-ch Relay Output Module
ADAM-4069	8-ch Relay Output Module
ADAM-4168	8-ch Relay Output Module

6.1.1.1.1. Read Coil Status

The Modbus Function 01 is used to read coil status, or the ON/OFF status of digital input (DI) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Variable: The variable that are mapped onto an input.

Mapping: The mapping status of each variable.

The data type of each channel is in single bit. If the value is “true”, it means that the channel is on; “false” for off. For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 digital inputs. This will require either 8 Boolean addresses or 1 Byte addresse.

Note!

Meaning of address expression:

% = Directly Mapped variable

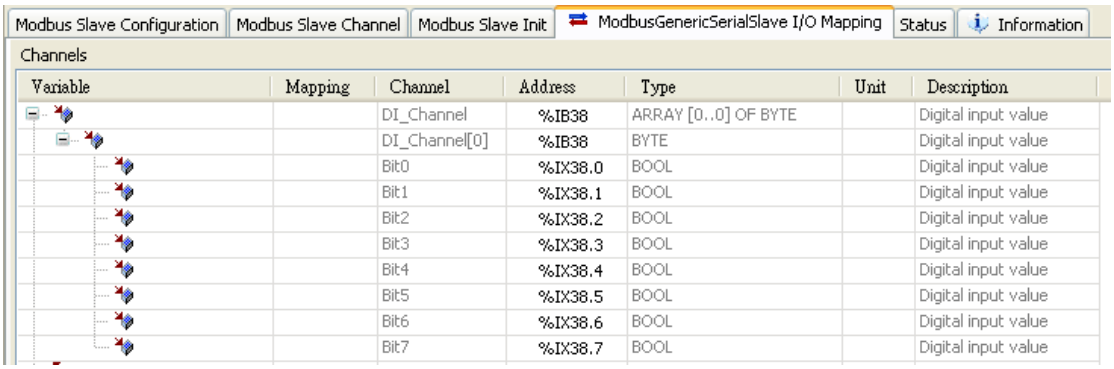
I = Physical Input

X = Single bit

$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.



6.1.1.1.2. Read Holding Registers

The Modbus Function 03 is used to read holding registers, or the quantity of registers to read from the analog input (AI) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Variable: The variable that are mapped onto an input.

Mapping: The mapping status of each variable.

The data type of each channel is in WORD.

For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 analog inputs. This will require 8 WORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

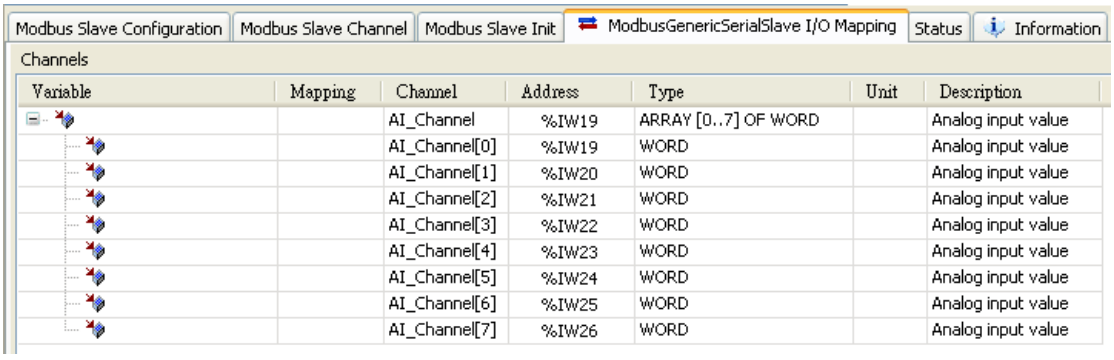
I = Physical Input

W = Single word (16 Bits)

$(N) = The starting address.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		AI_Channel	%IW19	ARRAY [0..7] OF WORD		Analog input value
		AI_Channel[0]	%IW19	WORD		Analog input value
		AI_Channel[1]	%IW20	WORD		Analog input value
		AI_Channel[2]	%IW21	WORD		Analog input value
		AI_Channel[3]	%IW22	WORD		Analog input value
		AI_Channel[4]	%IW23	WORD		Analog input value
		AI_Channel[5]	%IW24	WORD		Analog input value
		AI_Channel[6]	%IW25	WORD		Analog input value
		AI_Channel[7]	%IW26	WORD		Analog input value

6.1.1.1.3. Force Multiple Coils

The Modbus Function 15 is used to force multiple coils or ON/OFF state to digital output (DO) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Variable: The variable that are mapped onto an output.

Mapping: The mapping status of each variable.

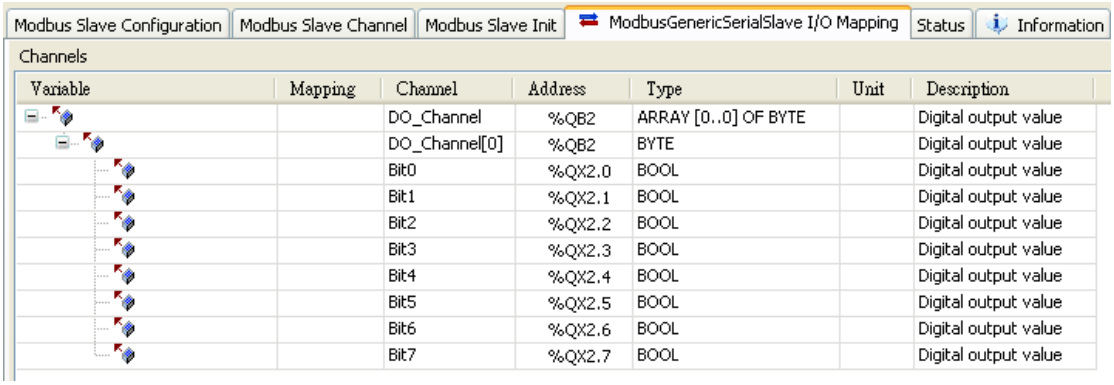
The data type of each channel is in single bit. Set the value to “true” for switching on the channel; “false” for switching off. For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 8 digital outputs. This will require either 8 Boolean addresses or 1 Byte address.

> ***Note!***
>
> Meaning of address expression:
>
> **%** = Directly Mapped variable
>
> **Q** = Physical Output
>
> **X** = Single bit
>
> **$(N1). $(N2)** = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.



Variable	Mapping	Channel	Address	Type	Unit	Description
		DO_Channel	%QB2	ARRAY [0..0] OF BYTE		Digital output value
		DO_Channel[0]	%QB2	BYTE		Digital output value
		Bit0	%QX2.0	BOOL		Digital output value
		Bit1	%QX2.1	BOOL		Digital output value
		Bit2	%QX2.2	BOOL		Digital output value
		Bit3	%QX2.3	BOOL		Digital output value
		Bit4	%QX2.4	BOOL		Digital output value
		Bit5	%QX2.5	BOOL		Digital output value
		Bit6	%QX2.6	BOOL		Digital output value
		Bit7	%QX2.7	BOOL		Digital output value

6.1.1.1.4. Preset Multiple Registers

The Modbus Function 16 is used to preset multiple register or write the contents to analog output (AO) modules. Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. In **ModbusGenericSerialSlave I/O Mapping**, it shows the I/O mapping status between variable to module channel. It consists of seven columns.

Variable: The variable that are mapped onto an output.

Mapping: The mapping status of each variable.

The data type of each channel is in WORD.
For detailed variable mapping information, see chapter 4.2.

Address: The starting physical address of the variables for this I/O group. The board shown below has 4 analog outputs. This will require either 4 WORD addresses.

Note!

Meaning of address expression:

% = Directly Mapped variable

Q = Physical Output

W = Single word (16 Bits)

$(N1). $(N2) = The starting address. The first number means the starting byte; the second number means the starting bit.

Type: The data type of each variable.

Description: The description of each variable.

		AO_Channel	%QW19	ARRAY [0..3] OF WORD		Analog output value
		AO_Channel[0]	%QW19	WORD		Analog output value
		AO_Channel[1]	%QW20	WORD		Analog output value
		AO_Channel[2]	%QW21	WORD		Analog output value
		AO_Channel[3]	%QW22	WORD		Analog output value

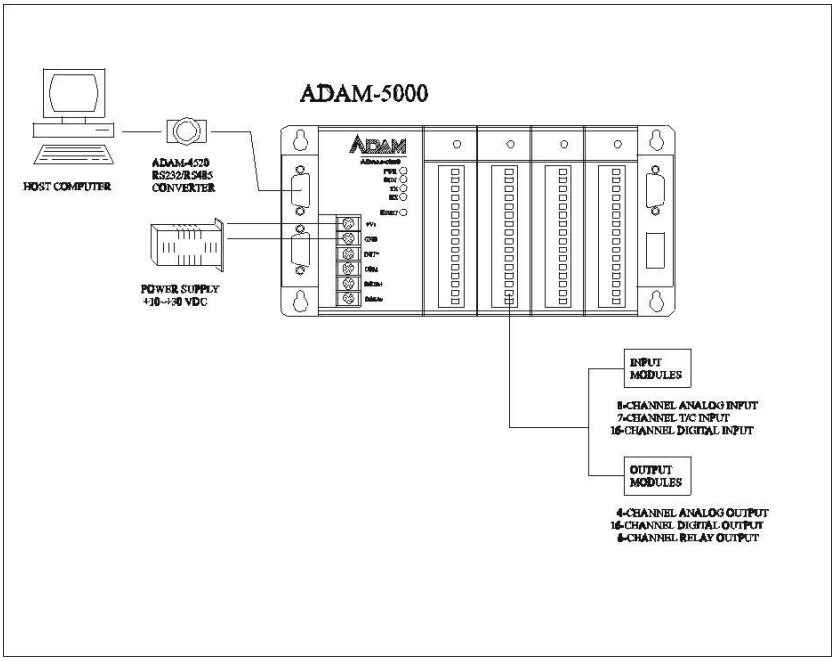
Note!

Please map the WORD variable onto an analog output channel. Do not use the BOOL variable onto a bit of an analog output channel.

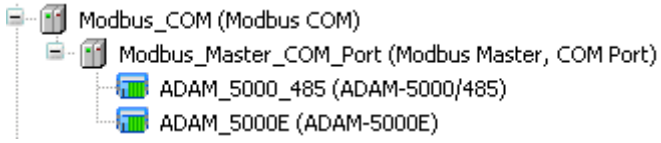
6.1.1.2. ADAM-5000 Series

The ADAM-5000 series is a complete product line that provides a wide variety of features in a data acquisition and control application. It includes 4 I/O-slots **ADAM-5000/485** and 8 I/O-slots **ADAM-5000E**. They are remotely controlled by the host computer through a set of commands and transmitted in a RS-485 / RS-232 network.

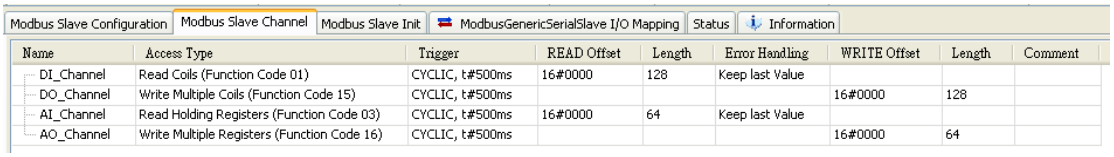
The following diagram shows the system configurations possible with the ADAM-5000.



Please refer to Chapter 5.1.1. and add **ADAM-5000/485 (4-slot)** or **ADAM-5000E (8-slot)** to device list.



Open the **Modbus Device editor** by double-clicking on the device icon in the device tree. The editor is subdivided in the 6 tabs. For more detailed information about **Modbus Slave Configuration** and **Modbus Slave Channel**, please refer to Chapter 5.1.1.



For ADAM-5000 series, Modbus slave address has been automatically assigned. CODESYS create default channel setting for different type module. The Modbus address mapping tables are shown below.

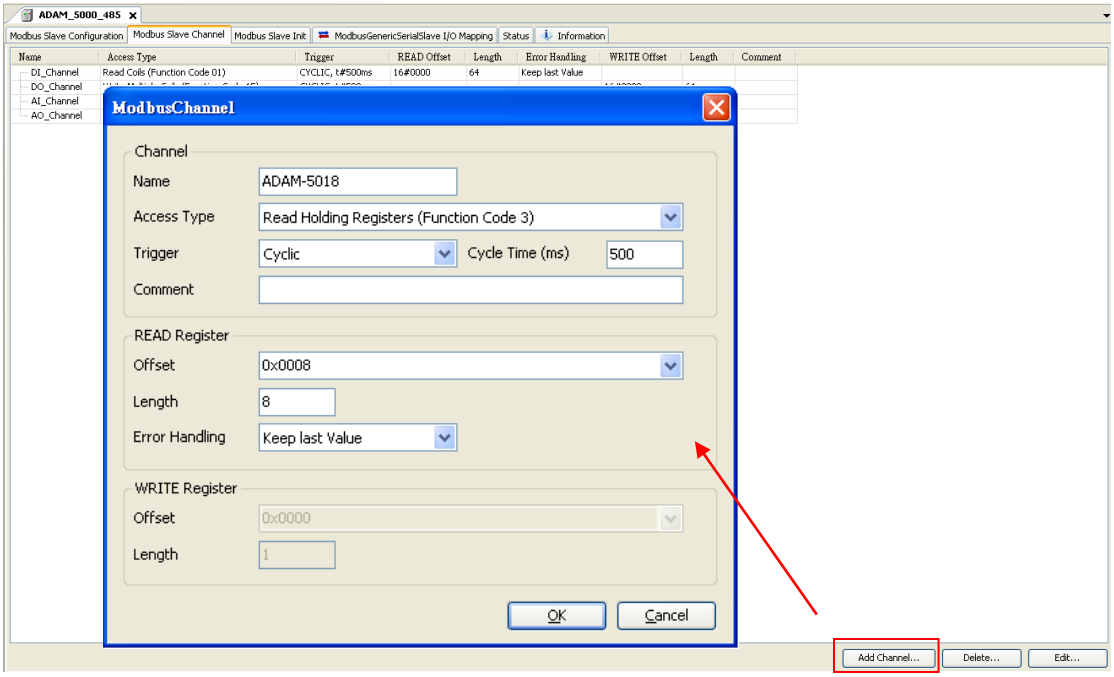
For Digital Input / Output Module:

	Slot 0	**Slot 1**	**Slot 2**	**Slot 3**	**Slot 4**	**Slot 5**	**Slot 6**	**Slot 7**
Bit 0	00001	00017	00033	00049	00065	00081	00097	00113
Bit 1	00002	00018	00034	00050	00066	00082	00098	00114
Bit 2	00003	00019	00035	00051	00067	00083	00099	00115
Bit 3	00004	00020	00036	00052	00068	00084	00100	00116
Bit 4	00005	00021	00037	00053	00069	00085	00101	00117
Bit 5	00006	00022	00038	00054	00070	00086	00102	00118
Bit 6	00007	00023	00039	00055	00071	00087	00103	00119
Bit 7	00008	00024	00040	00056	00072	00088	00104	00120
Bit 8	00009	00025	00041	00057	00073	00089	00105	00121
Bit 9	00010	00026	00042	00058	00074	00090	00106	00122
Bit 10	00011	00027	00043	00059	00075	00091	00107	00123
Bit 11	00012	00028	00044	00060	00076	00092	00108	00124
Bit 12	00013	00029	00045	00061	00077	00093	00109	00125
Bit 13	00014	00030	00046	00062	00078	00094	00110	00126
Bit 14	00015	00031	00047	00063	00079	00095	00111	00127
Bit 15	00016	00032	00048	00064	00080	00096	00112	00128

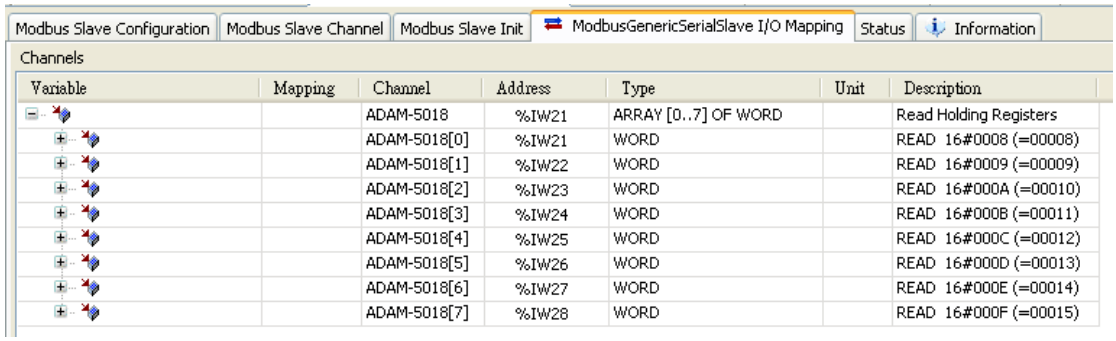
For Analog Input / Output Module:

	Slot 0	Slot 1	Slot 2	Slot 3	Slot 4	Slot 5	Slot 6	Slot 7
Ch 0	40001	40009	40017	40025	40033	40041	40049	40057
Ch 1	40002	40010	40018	40026	40034	40042	40050	40058
Ch 2	40003	40011	40019	40027	40035	40043	40051	40059
Ch 3	40004	40012	40020	40028	40036	40044	40052	40060
Ch 4	40005	40013	40021	40029	40037	40045	40053	40061
Ch 5	40006	40014	40022	40030	40038	40046	40054	40062
Ch 6	40007	40015	40023	40031	40039	40047	40055	40063
Ch 7	40008	40016	40024	40032	40040	40048	40056	40064

You can use default channel setting or add a new channel to a Modbus slave. For example, you insert ADAM-5018 (Analog Input module) in the second slot (slot 1) of ADAM-5000E. According to the Modbus address mapping table, start address as 40009 and length as 8. Now, go to **Modbus Slave Channel** page and click **Add Channel.** Select **Access type** as **Read Holding Register** and fill in **Offset** and **Length.** This dialog may be closed by clicking **OK** for creating the channel.



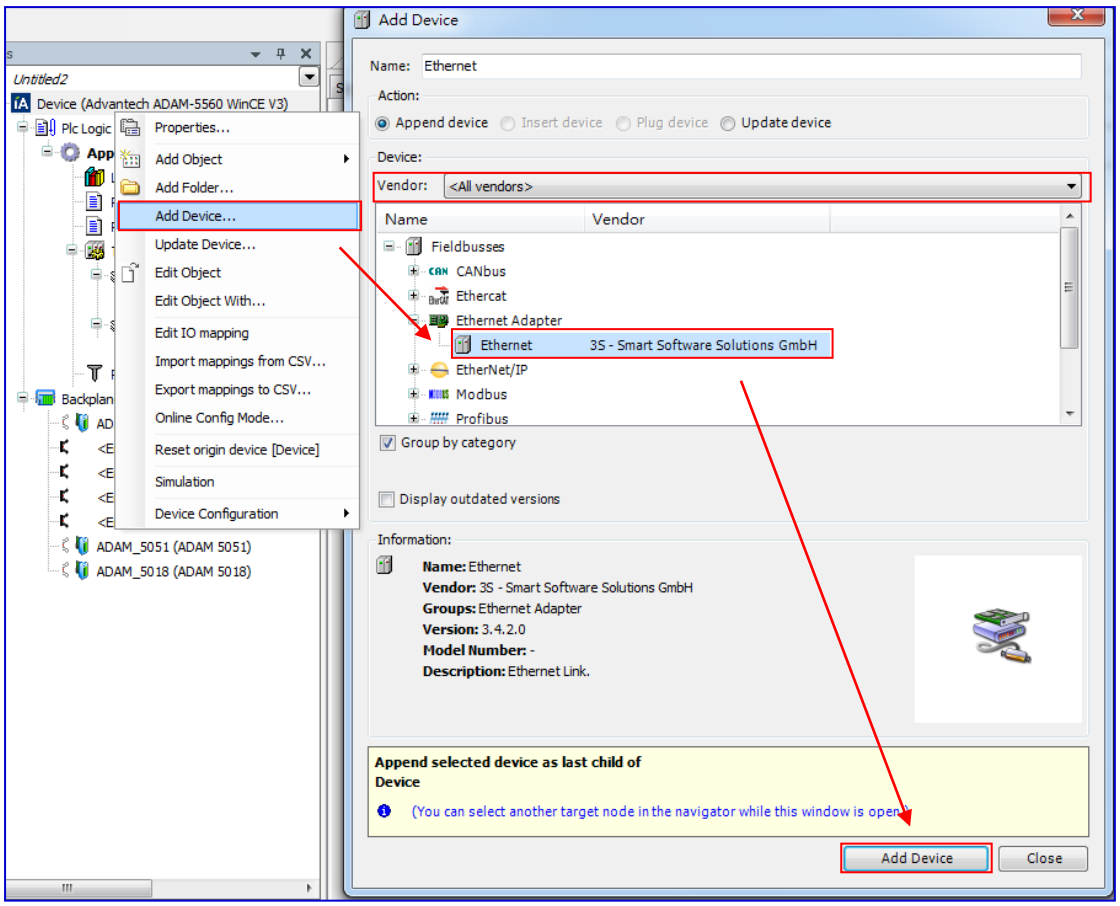
Correspondent to the set channels, the slave's process data can then be monitored under **ModbusGenericSerialSlave I/O Mapping** page.

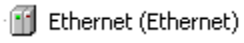


Variable	Mapping	Channel	Address	Type	Unit	Description
		ADAM-5018	%IW21	ARRAY [0..7] OF WORD		Read Holding Registers
		ADAM-5018[0]	%IW21	WORD		READ 16#0008 (=00008)
		ADAM-5018[1]	%IW22	WORD		READ 16#0009 (=00009)
		ADAM-5018[2]	%IW23	WORD		READ 16#000A (=00010)
		ADAM-5018[3]	%IW24	WORD		READ 16#000B (=00011)
		ADAM-5018[4]	%IW25	WORD		READ 16#000C (=00012)
		ADAM-5018[5]	%IW26	WORD		READ 16#000D (=00013)
		ADAM-5018[6]	%IW27	WORD		READ 16#000E (=00014)
		ADAM-5018[7]	%IW28	WORD		READ 16#000F (=00015)

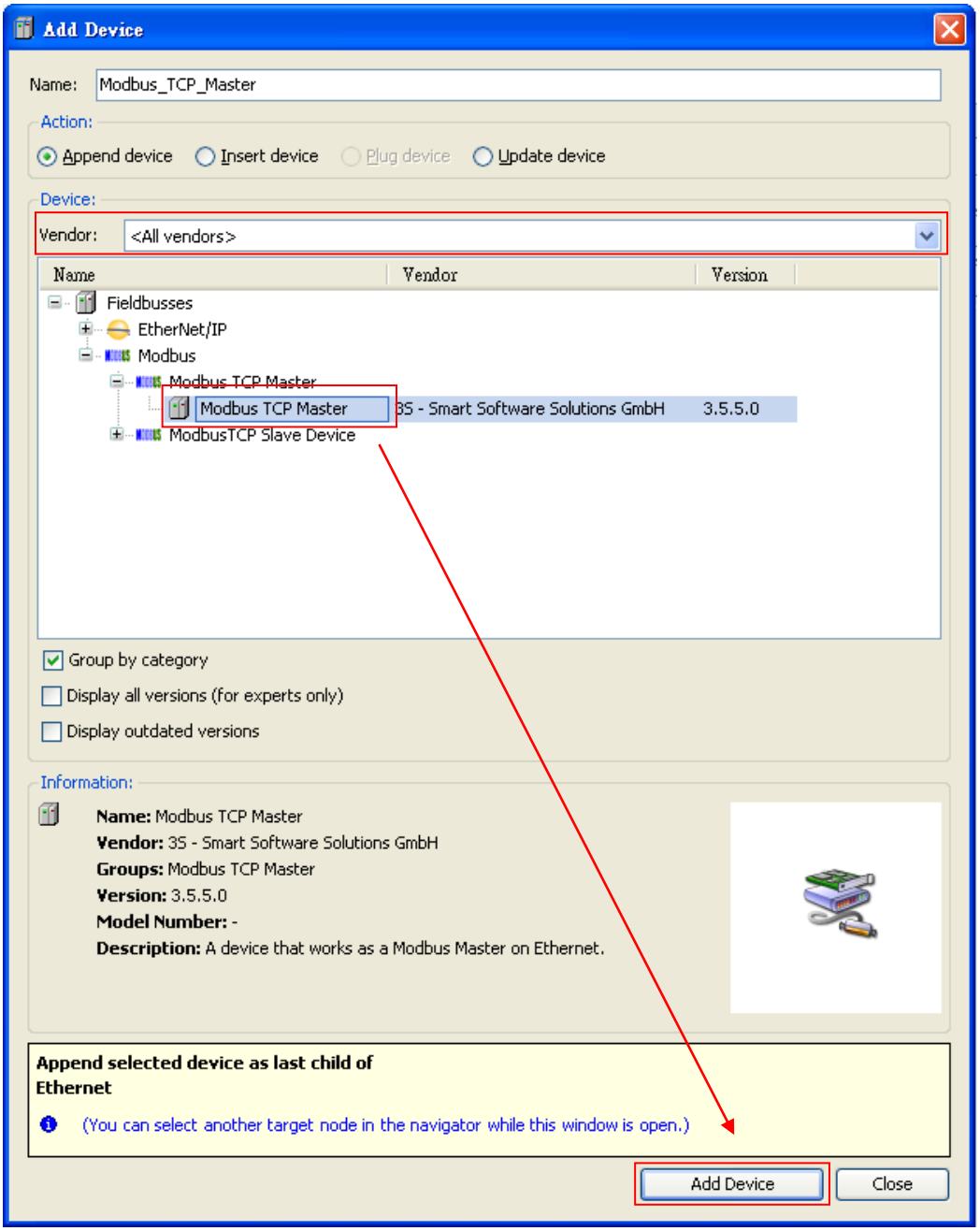
6.1.2. Modbus TCP Client

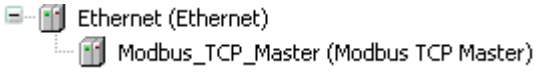
In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog. Choose **Ethernet** in **Ethernet Adapter** and click **Add Device** to close the dialog.



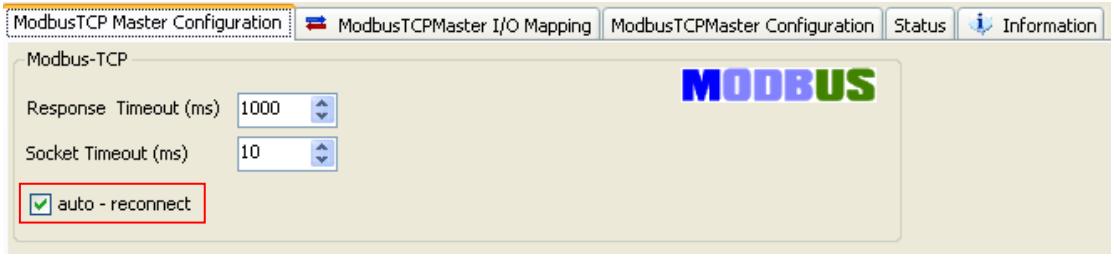
Right click on Ethernet adapter 

in the device tree and click **Add Device.** Type the device name in Name container. Choose **Modbus TCP Master** in the **Modbus** option. Click **Add Device** to proceed and then press **Close** to close the device dialog.

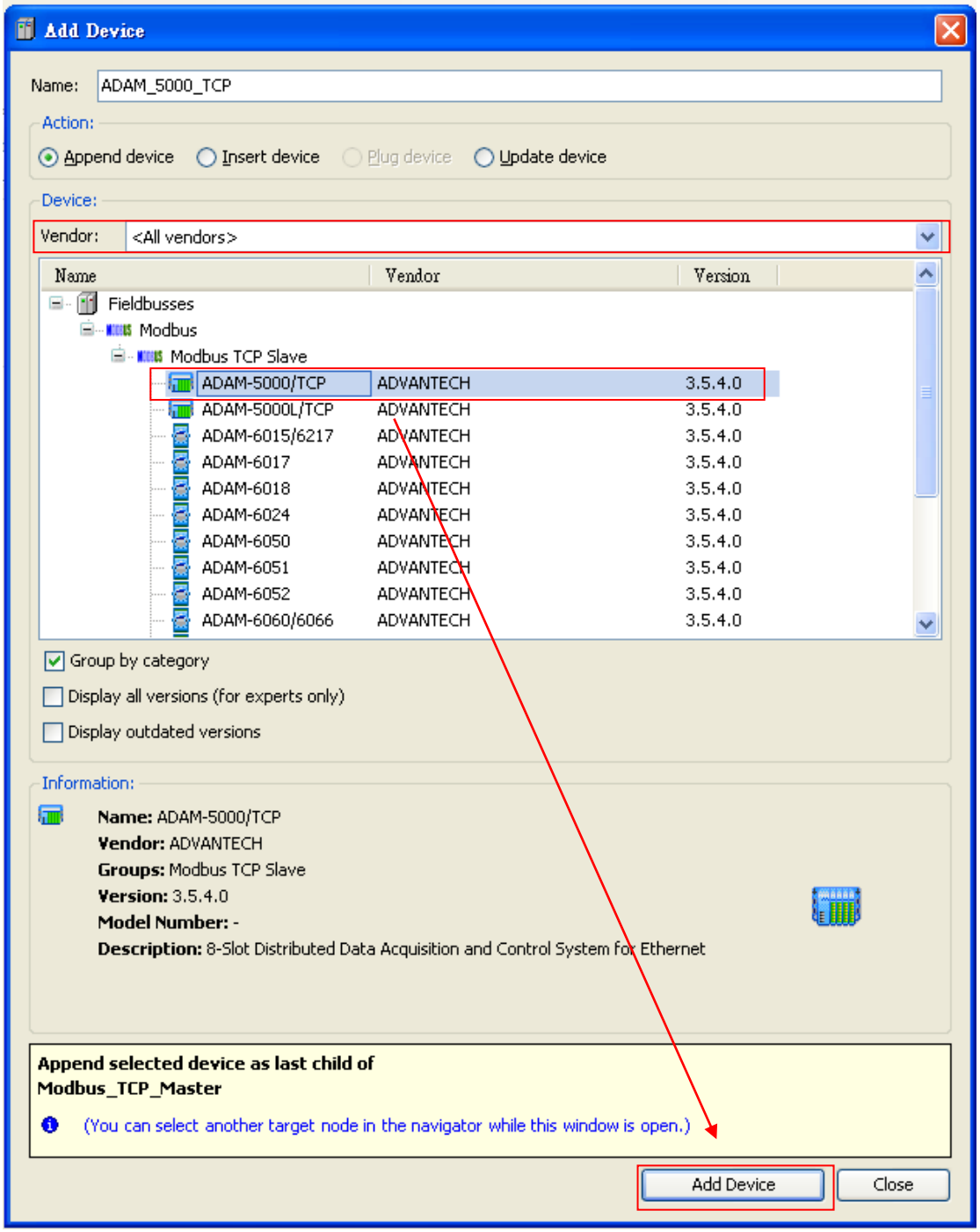


Now, you'll see Modbus TCP master 

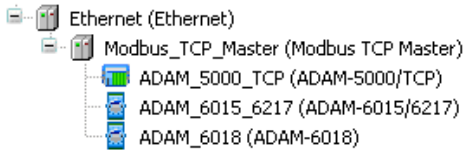
in the device tree. Double-click Modbus TCP master to set master configuration. For further convenience, recommend you to check **auto-restart communication**.



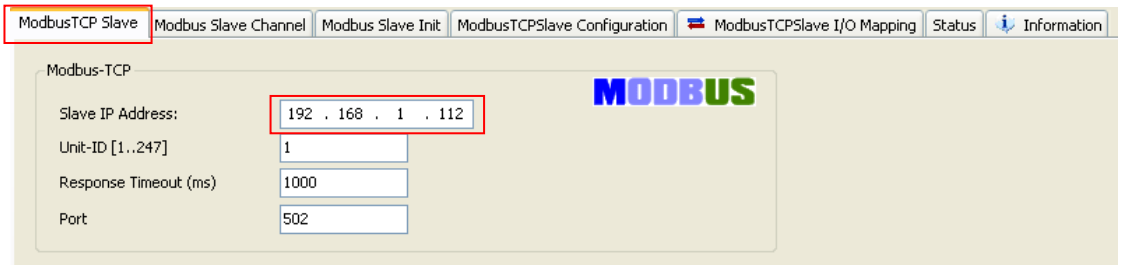
Right click on Modbus TCP master in the device tree and click **Add Device** in context menu. It will open the **Add Device dialog**, where you choose one available device for the current connection. Type the device name in Name container. Click **Add Device** to proceed and then press **Close** to close the device dialog.



Now, you will see Advantech ADAM modules on device list.

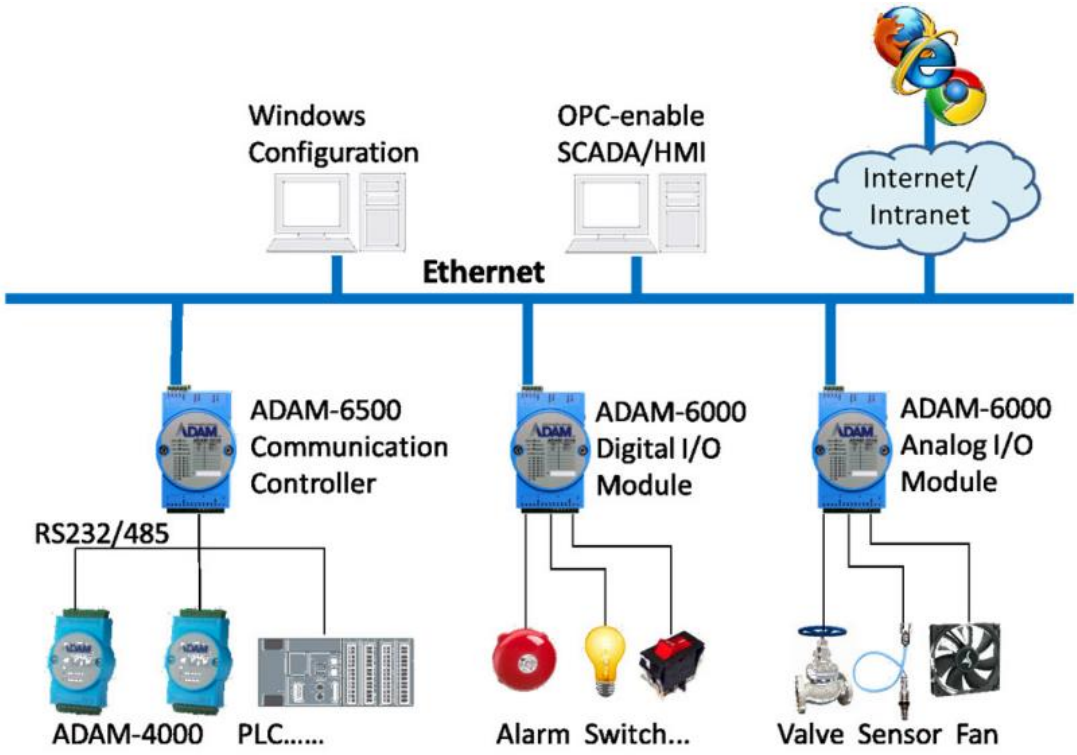


Now, set the IP address and node ID at ADAM module. Double-click on the device icon in the device tree and open the respective editors. Enter the module IP address and node ID in the register-tab "Modbus TCP Slave" (in this example: address 192.168.1.112 and node ID 1) and keep port as 502.



6.1.2.1. ADAM-6000 Series

ADAM-6000 Ethernet-based data acquisition and control modules provide I/O, data acquisitions, and networking in one module to build a cost-effective, distributed monitoring and control solution for a wide variety of applications. Through standard Ethernet networking, ADAM-6000 retrieves I/O values from sensors, and can publish them as a real-time I/O values to networking nodes via LAN, Intranet, or Internet.
The figure below shows the brief overview of the ADAM-6000 system architecture.



Advantech provides 16 types of ADAM-6000 modules for various applications so far. Following table is ADAM-6000 series support list.

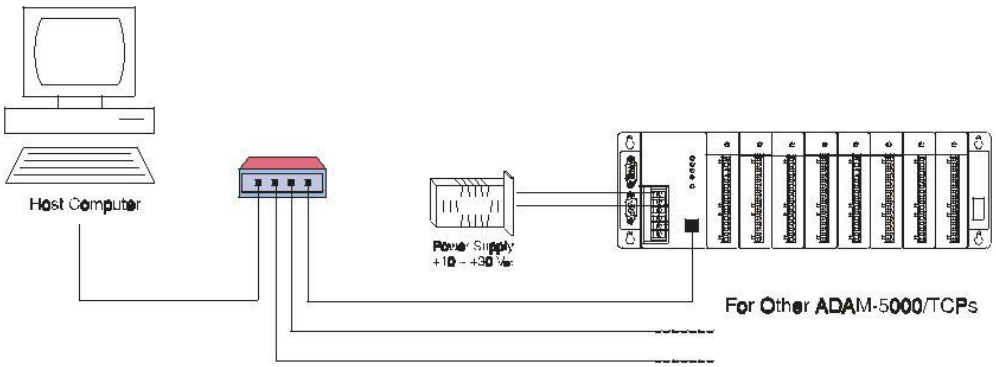
Name	Specification
ADAM-6015	7-ch RTD Input Module
ADAM-6217	8-ch Analog Input Module
ADAM-6017	8-ch Analog Input Module with 2-ch DO
ADAM-6018	8-ch Thermocouple Input Module with 8-ch DO
ADAM-6050	18-ch Digital I/O Module
ADAM-6051	14-ch Digital I/O Module with 2-ch Counter
ADAM-6052	16-ch Digital I/O Module
ADAM-6266	4-ch Relay Output Module with 4-ch DI
ADAM-6250	15-ch Digital I/O Module
ADAM-6060	6-ch Digital Input and 6-ch Relay Module
ADAM-6066	6-ch Digital Input and 6-ch Power Relay Module
ADAM-6024	12-ch Universal Input/Output Module
ADAM-6224	4-ch Analog Output Module
ADAM-6251	16-ch Digital Input Module
ADAM-6256	16-ch Digital Output Module
ADAM-6260	6-ch Relay Output Module

For more detailed information about how to control ADAM-6000 modules, please refer to Chapter 5.1.1.1.1 to Chapter 5.1.1.1.4.

6.1.2.2. ADAM-5000 Series

ADAM-5000/TCP Series works as a Modbus data server. It allows PCs or tasks to access its current data simultaneously from LAN, Intranet, or Internet. The ADAM-5000/TCP Series uses a convenient backplane system common to the ADAM-5000 series. Advantech's complete line of ADAM-5000 modules integrates with the ADAM-5000/TCP Series to support your applications.

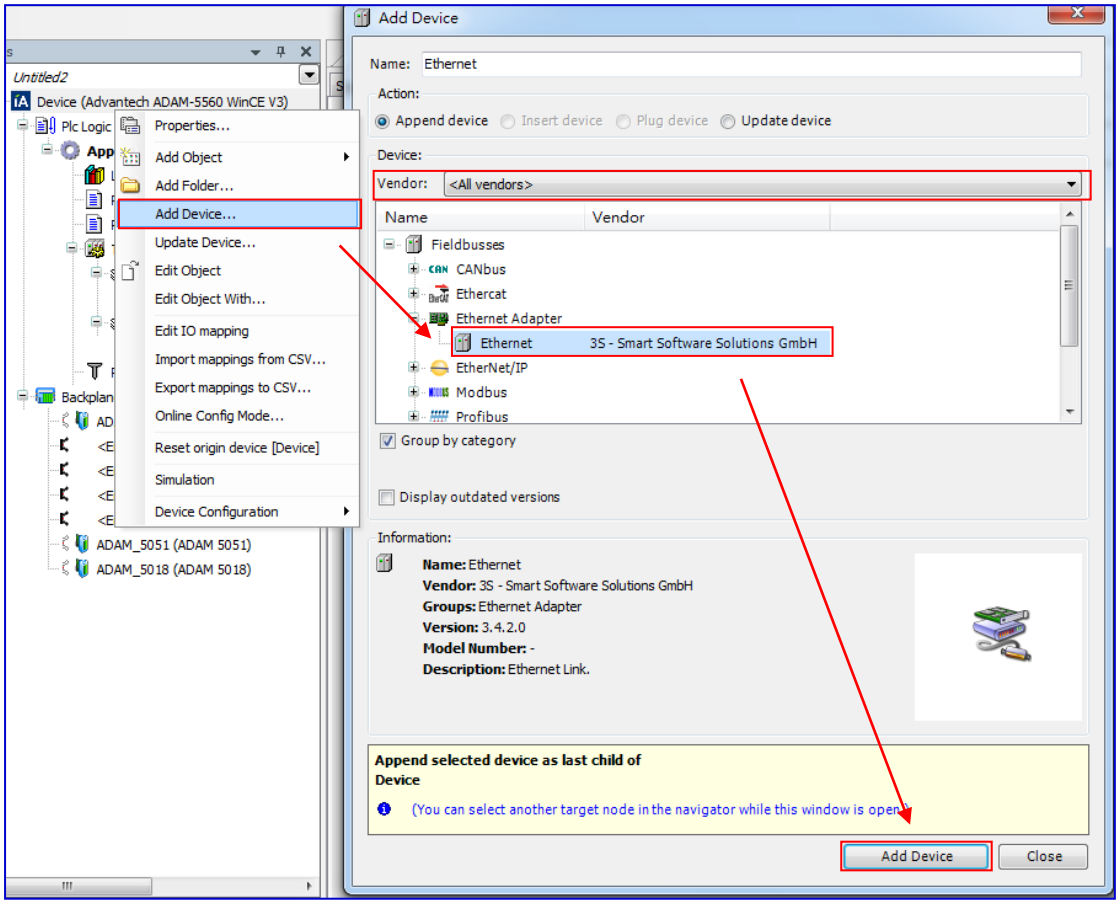
The figure below shows the brief overview of the ADAM-5000/TCP system architecture.

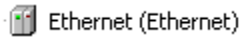


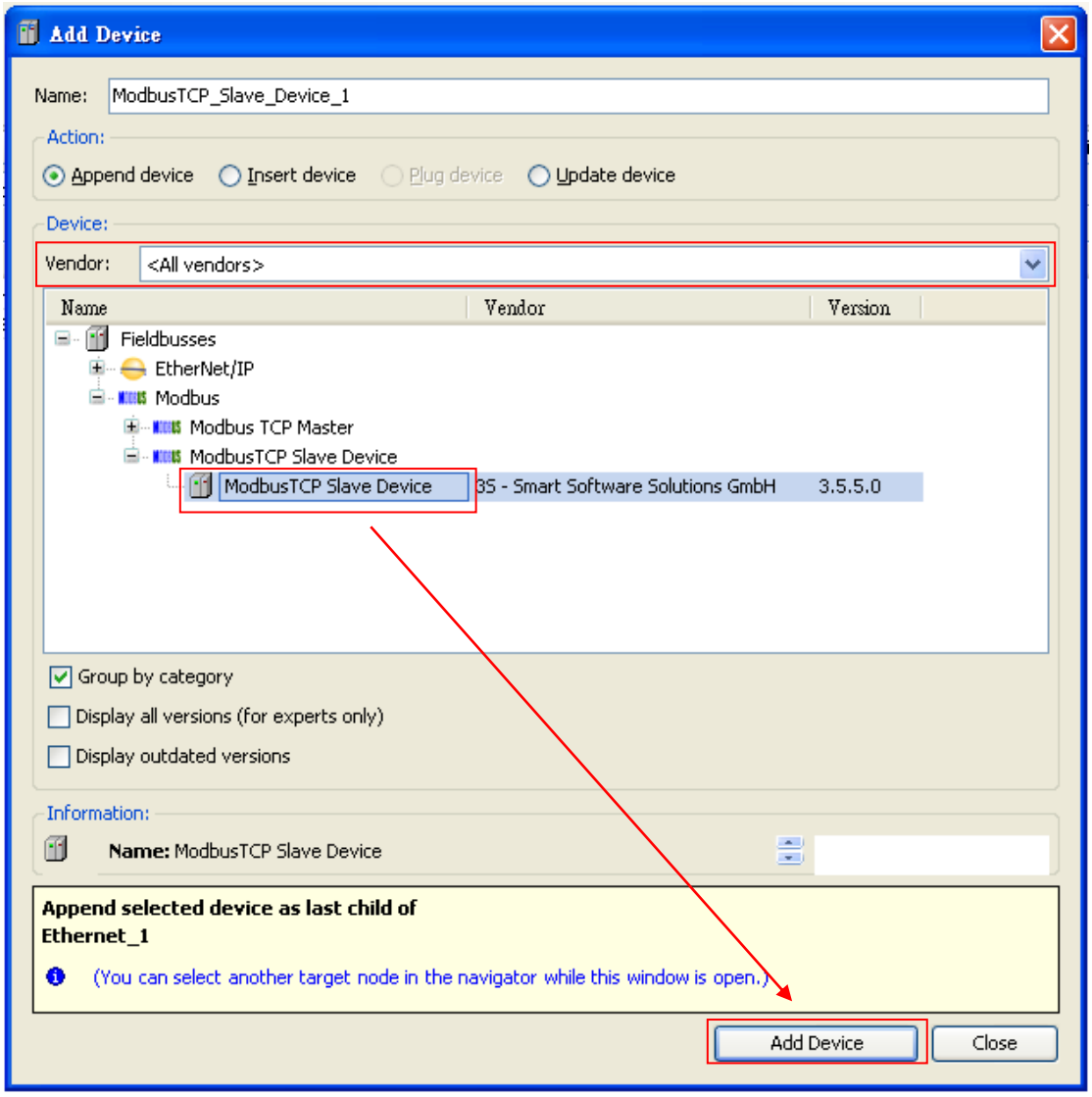
For more detailed information about how to access ADAM-5000 current data via Modbus protocol, please refer to Chapter 5.1.1.2.

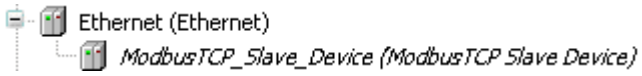
6.1.3. Modbus TCP Server

In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog. Choose **Ethernet** in **Ethernet Adapter** and click **Add Device** to close the dialog.



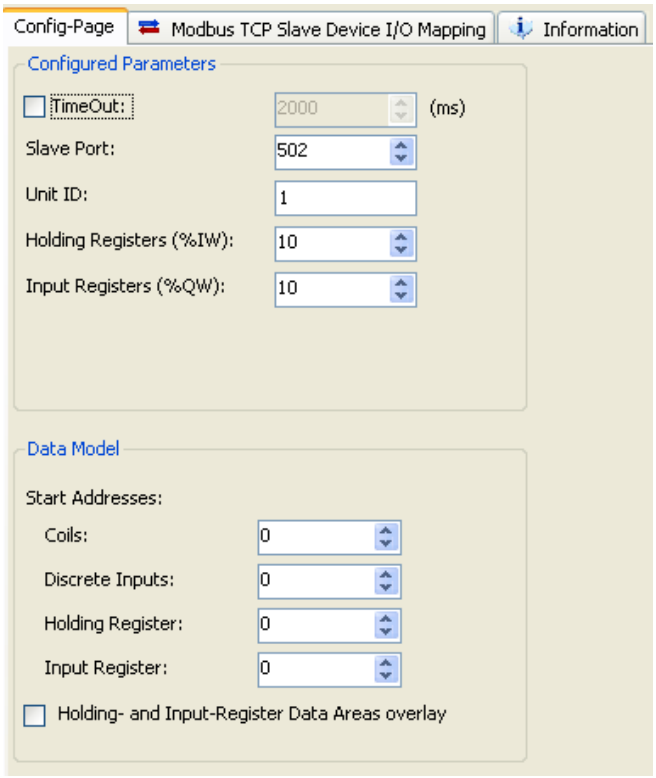
Right click on Ethernet adapter  in the device tree and click **Add Device.** Type the device name in Name container. Choose **Modbus TCP Slave** in the **Modbus** option. Click **Add Device** to proceed and then press **Close** to close the device dialog.



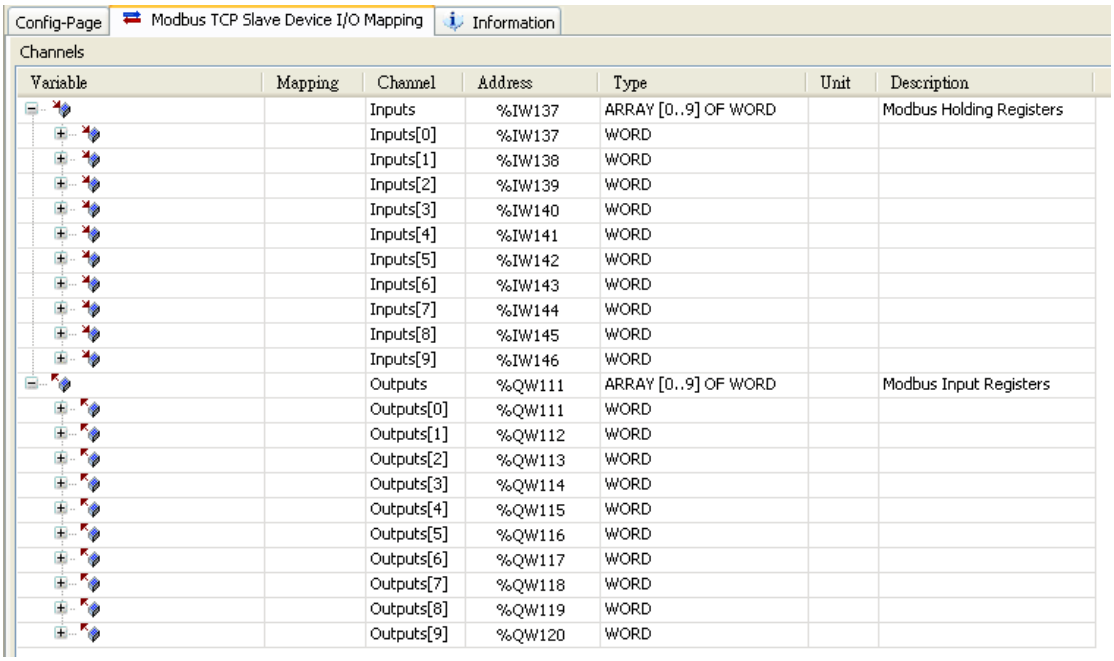
Now, you'll see Modbus TCP slave 

in the device tree. Double-click Modbus TCP slave and go to **Config-Page** to set configuration.

Timeout	Activation of the timing supervision function. The timeout interval is given in milliseconds. Values are adjustable in steps of 500 ms. If there is no write command received within this time, the outputs will be reset to 0.**To keep the output value, we recommend you to uncheck Timeout.**
Slave Port	Port number of the slave. Keep slave port as 502.
Unit-ID	(Optional) Unit ID of the slave.
Holding Register (%IW)	Number of holding register: Possible values: 2-500.
Input Register (%QW)	Number of input register: Possible values: 2-500.



Correspondent to the number of holding register and input register, the slave channel can then be mapped under **Modbus TCP Slave Device I/O Mapping** page. For more detailed information about how to map variable, please refer to Chapter 4.3..



Variable	Mapping	Channel	Address	Type	Unit	Description
		Inputs	%IW137	ARRAY [0..9] OF WORD		Modbus Holding Registers
		Inputs[0]	%IW137	WORD		
		Inputs[1]	%IW138	WORD		
		Inputs[2]	%IW139	WORD		
		Inputs[3]	%IW140	WORD		
		Inputs[4]	%IW141	WORD		
		Inputs[5]	%IW142	WORD		
		Inputs[6]	%IW143	WORD		
		Inputs[7]	%IW144	WORD		
		Inputs[8]	%IW145	WORD		
		Inputs[9]	%IW146	WORD		
		Outputs	%QW111	ARRAY [0..9] OF WORD		Modbus Input Registers
		Outputs[0]	%QW111	WORD		
		Outputs[1]	%QW112	WORD		
		Outputs[2]	%QW113	WORD		
		Outputs[3]	%QW114	WORD		
		Outputs[4]	%QW115	WORD		
		Outputs[5]	%QW116	WORD		
		Outputs[6]	%QW117	WORD		
		Outputs[7]	%QW118	WORD		
		Outputs[8]	%QW119	WORD		
		Outputs[9]	%QW120	WORD		

6.2. CANOpen

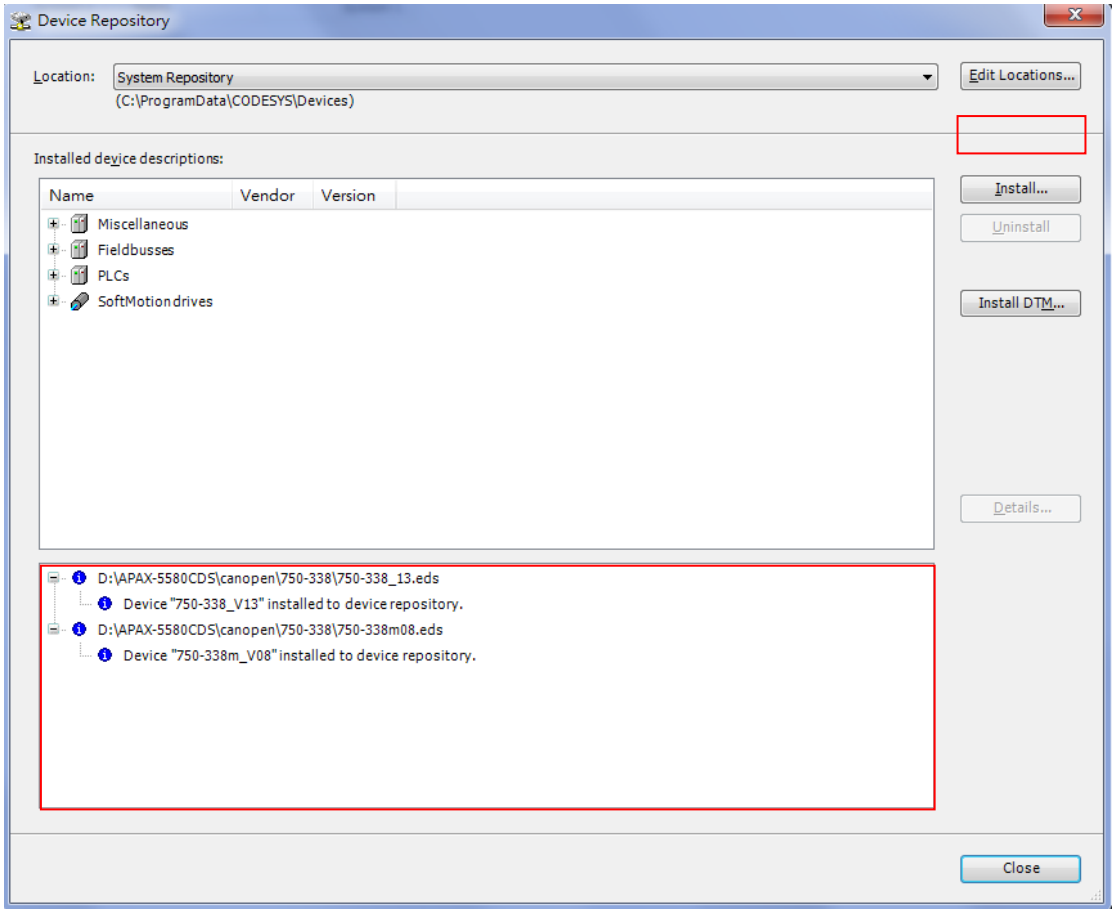
6.2.1. CANOpen Client

6.2.1.1. Configuration Files Installation

Before connecting to slaves by using CANOpen client, please install the related EDS files (*.eds) first.

Start CODESYS and perform command **Device Repository**  from the menu (**Tools -> Device Repository**). Click “Install” and then select the related EDS file you want to install.

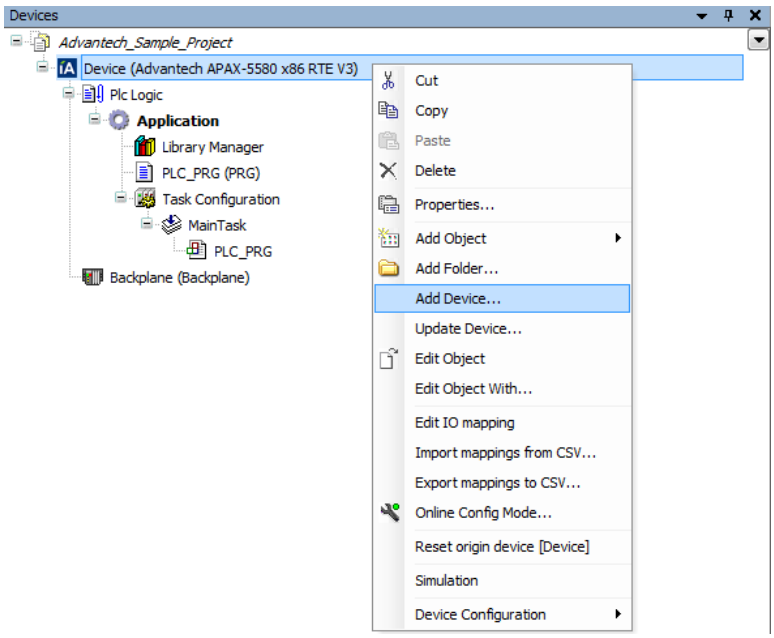
After installed successfully, the following information will be shown: “Device xxxx installed to device repository”.



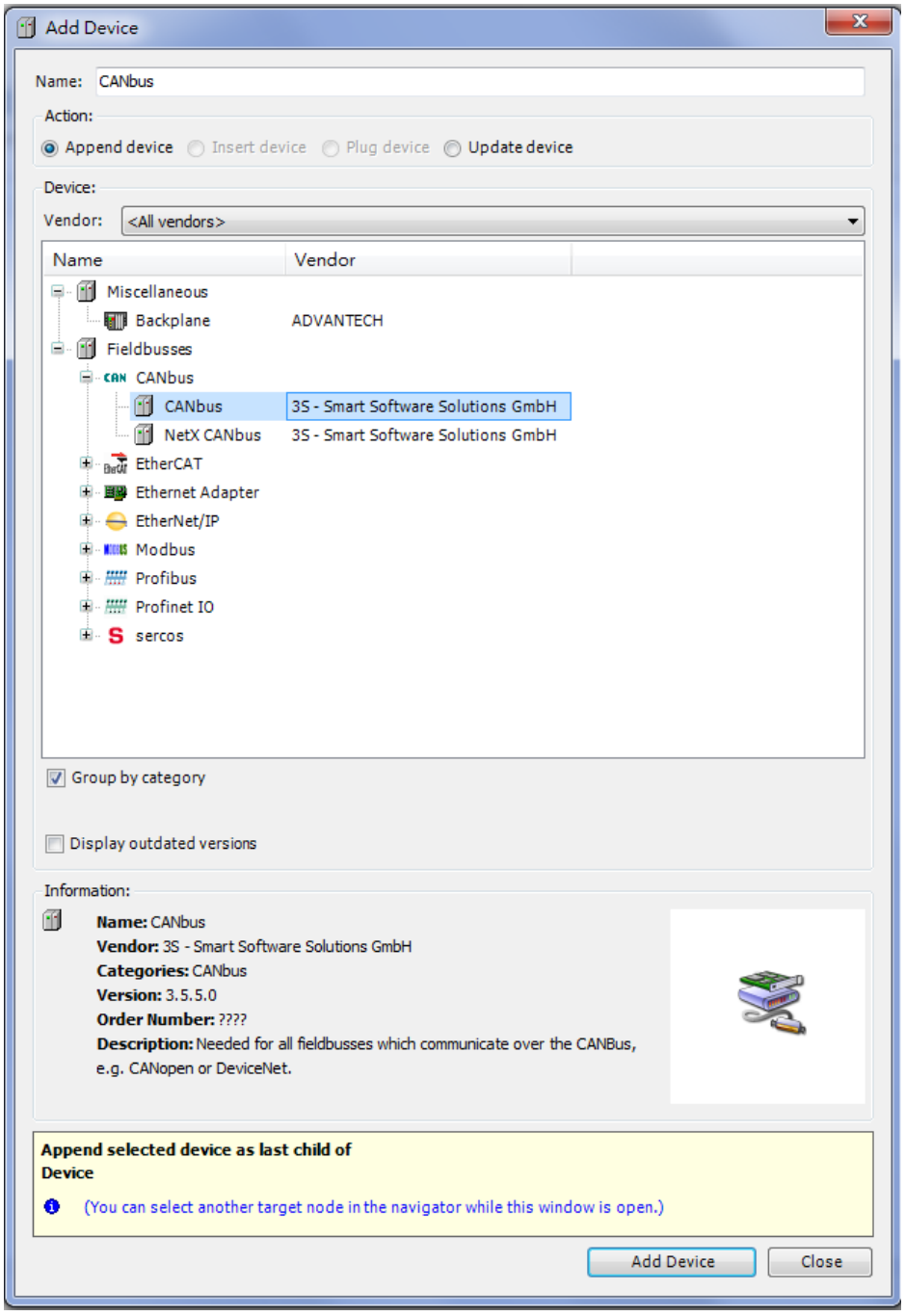
6.2.1.2. Scan for Slaves

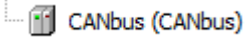
First, connect to your specified Advantech X86 RTE platform. Please refer to Chapter 3.4.

In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog.

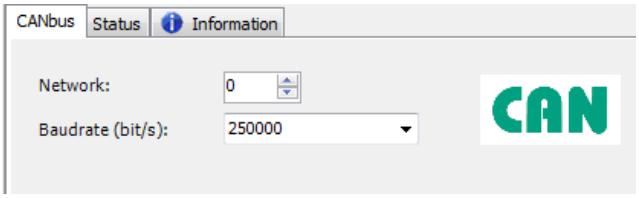


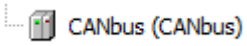
Choose **CANbus** in the **CANbus** option (**Fieldbusses -> CANbus**) and click **Add Device** to proceed and then press Close to close the device dialog.



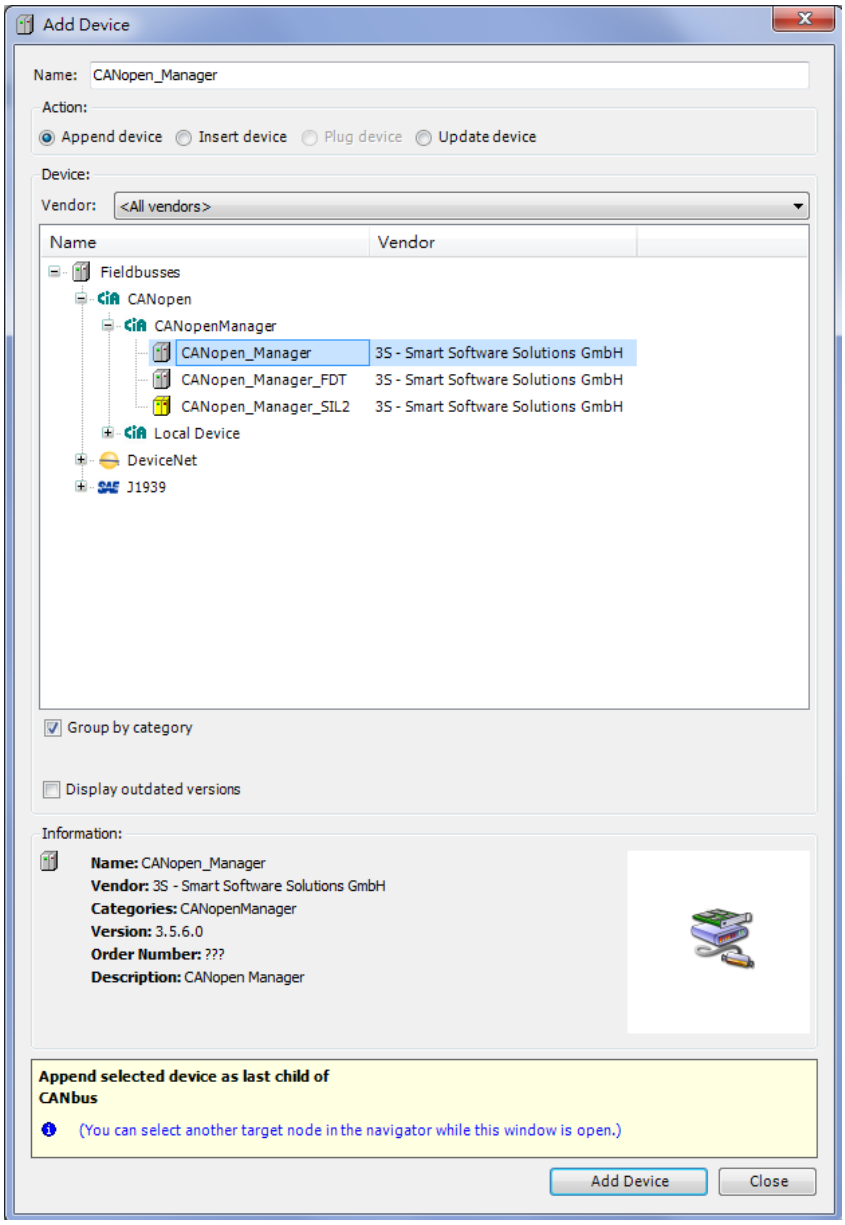
Now, you'll see CANbus 

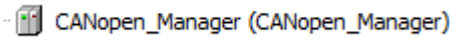
in the device tree. Double-click the CANbus icon to set configuration. Set up the network ID for the CAN port and the corresponding baud rate.



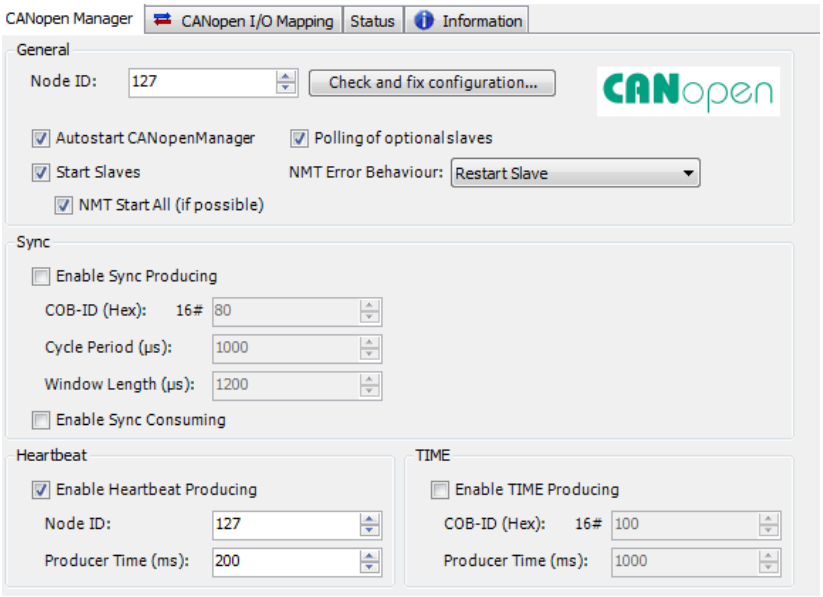
Right click on CANbus  in the device tree and click **Add Device.**

Choose **CANOpen_Manager** in the **CANOpenManager** option (**Fieldbussed -> CANOpen -> CANOpenManager**). Click **Add Device** to proceed and then press **Close** to close the device dialog.



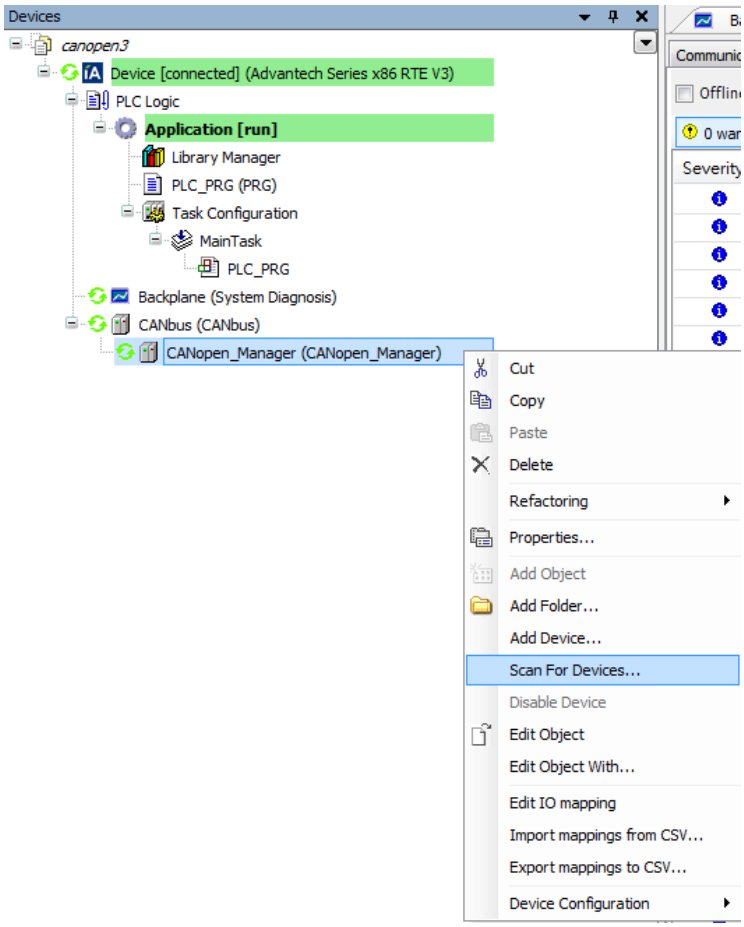
Now, you'll see CANOpen_Manager 

in the device tree. Double-click the CANOpen_Manager icon to set configuration.

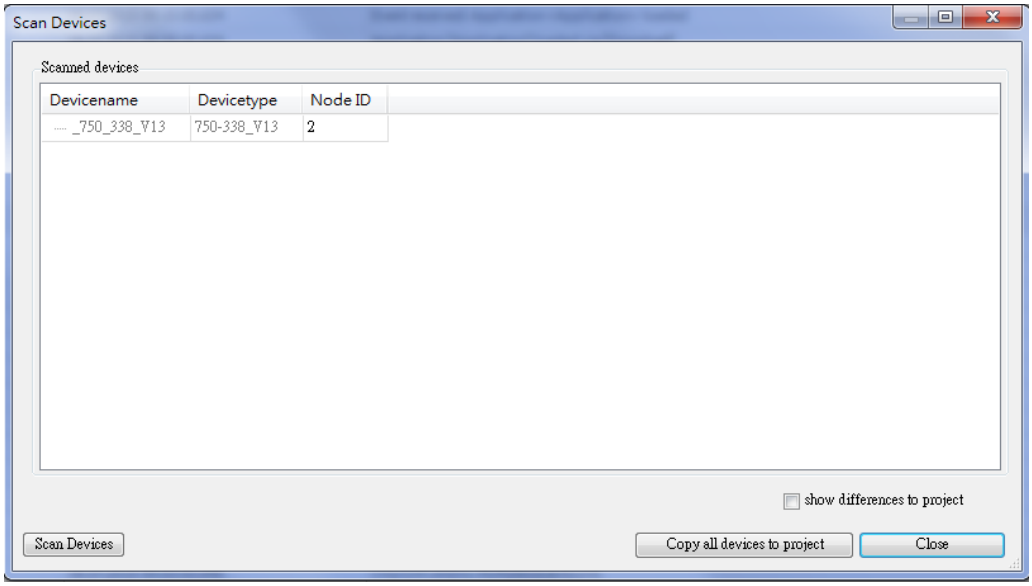


At the first scan, at least **once a login (and running)** must have been done. Otherwise, the Advantech X86 RTE platform must be running before a scan.

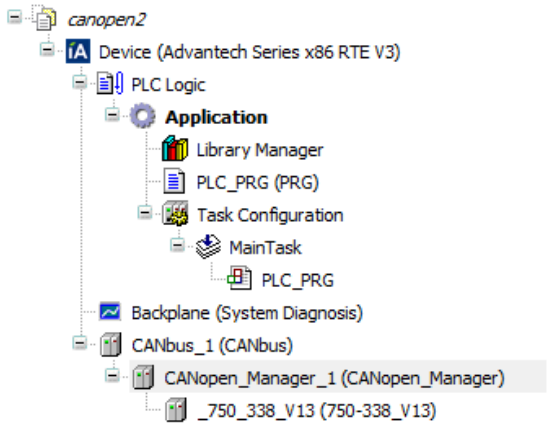
Choose the **CANOpen_Manager** and right click **Scan For Devices** in context menu.



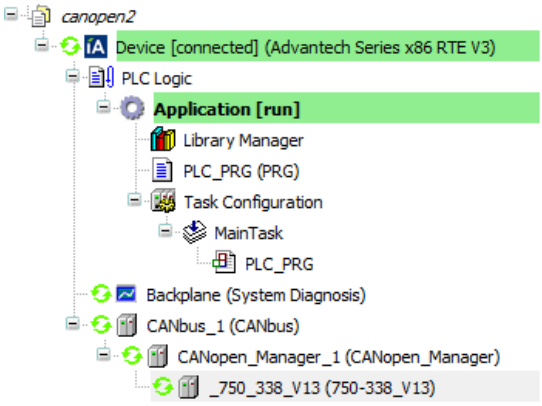
A list of all devices and modules are found during the last scan. Select the specified one device and then copy to the project. Or copy all listed devices to the project.



Now, you'll see the devices copied to the project under the **CANOpen_Manager**.



After completely finish device configuration, it is necessary to login again.



6.3. EtherNet/IP

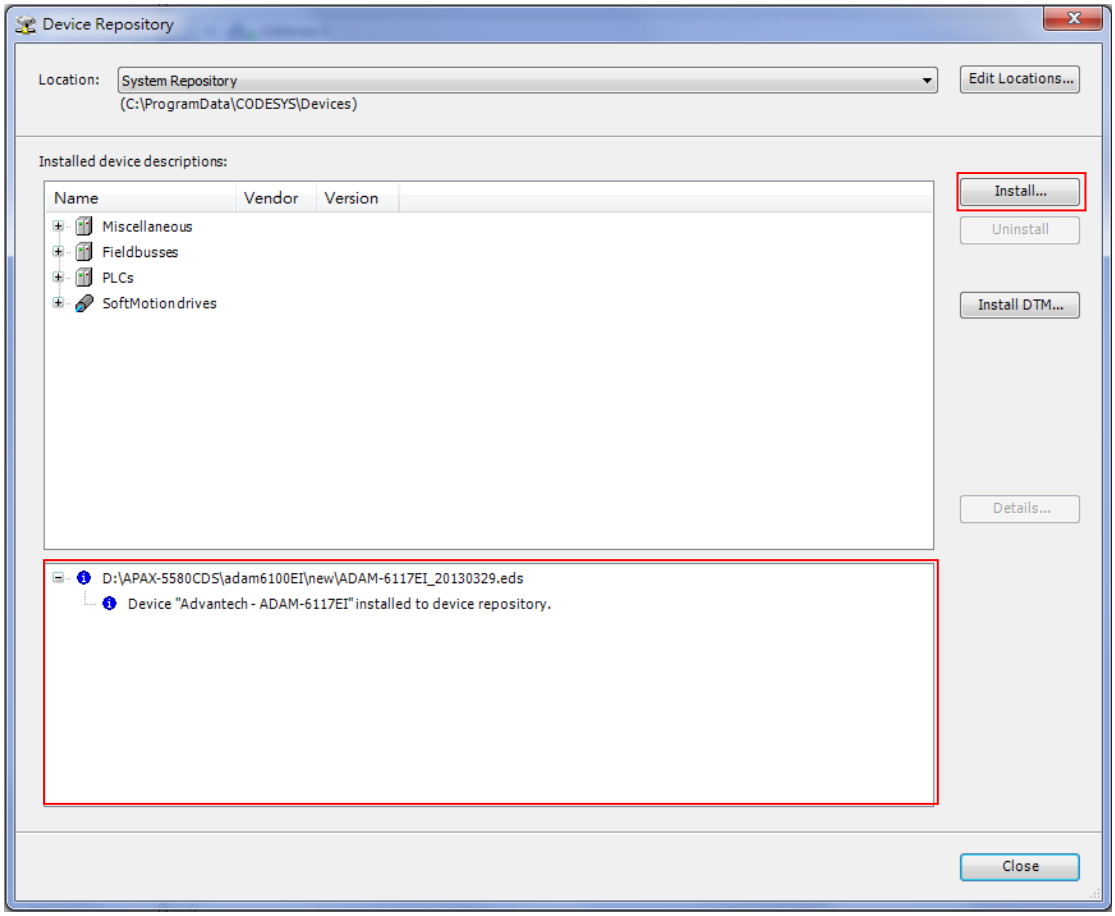
6.3.1. EtherNet/IP Client

6.3.1.2. Configuration Files Installation

Before connecting to slaves by using Profinet client, please install the related EDS files (*.eds) first.

Start CODESYS and perform command **Device Repository**  from the menu (**Tools -> Device Repository**). Click “Install” and then select the related EDS file you want to install.

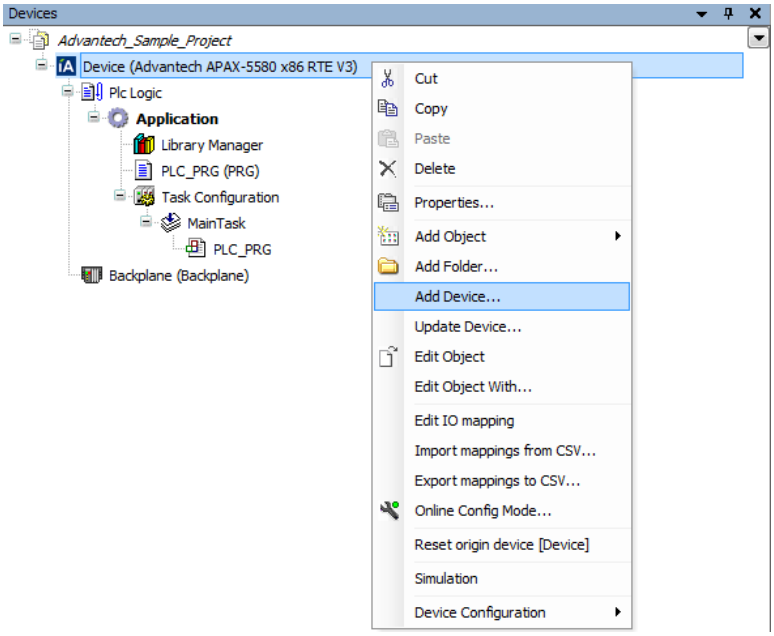
After installed successfully, the following information will be shown: “Device xxxx installed to device repository”.



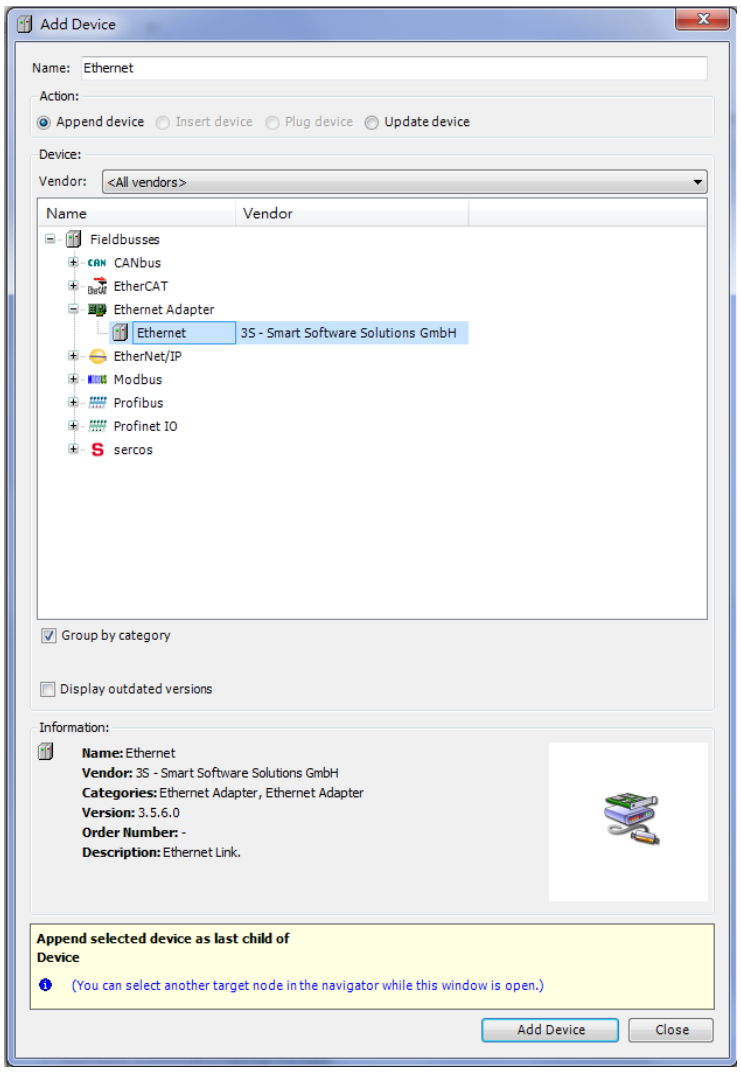
6.3.1.3. Add Slaves

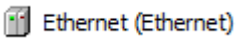
First, connect to your specified Advantech X86 RTE platform. Please refer to Chapter 3.4.

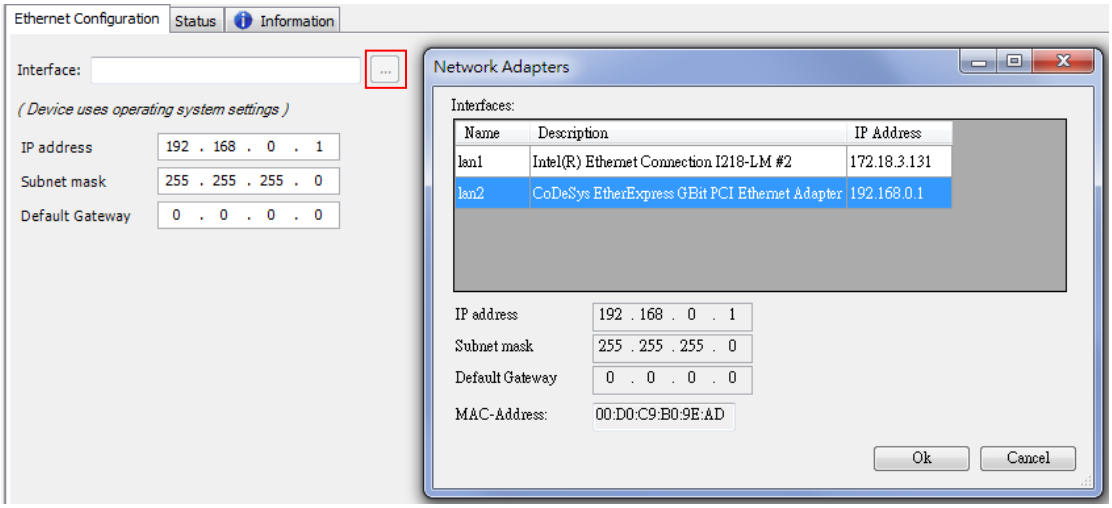
In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog.



Choose **Ethernet** in the **Ethernet Adapter** option and click **Add Device** to proceed and then press Close to close the device dialog.



Now, you'll see Ethernet  in the device tree. Double-click the Ethernet icon to set configuration. Click **Interface** and then select **any one Ethernet Adapter.**

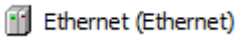


Note!

(1) Both LAN#1 and LAN#2 can be used for EtherNet/IP.

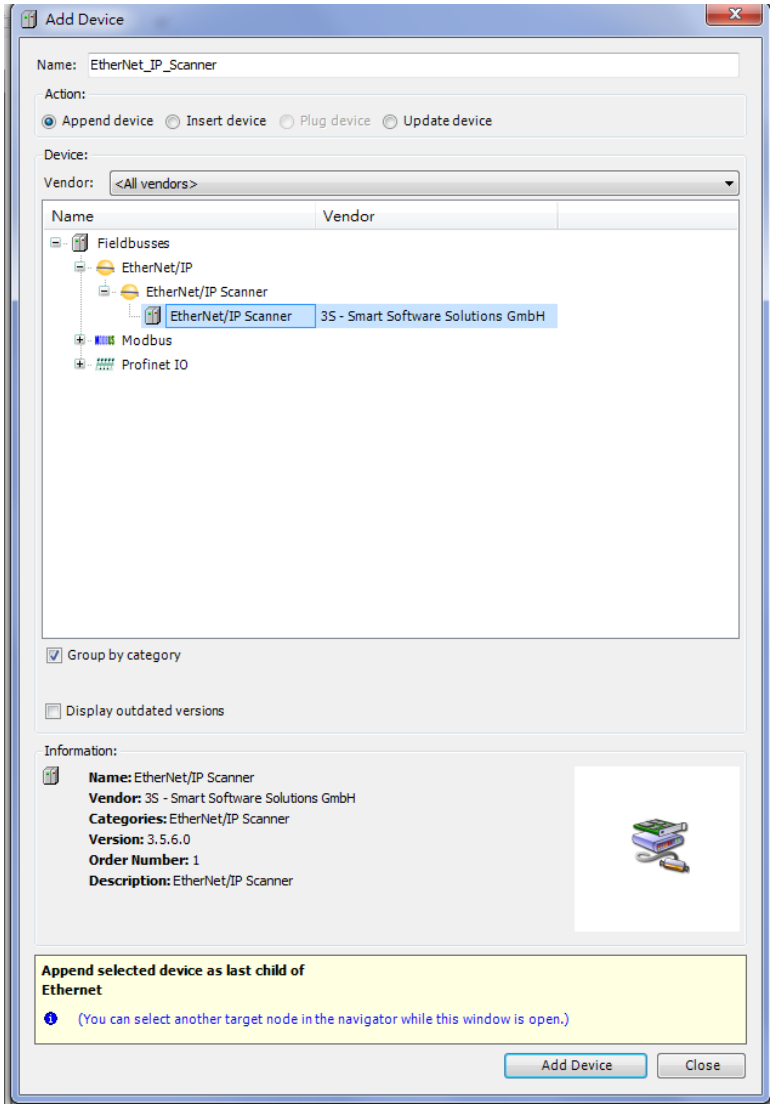
(2) The IP address should be setup completely by using “Change Adapter settings” in Advantech X86 RTE platform.

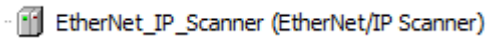
(3) The Ethernet Adapter must not have a Unicode String Name. Please rename your adapter using ASCII characters only.

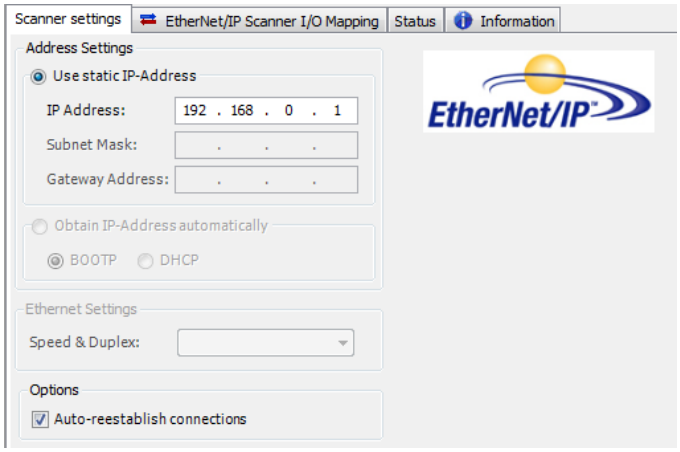
Right click on Ethernet 

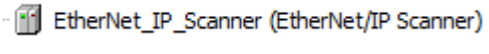
in the device tree and click **Add Device.**

Choose **EtherNet/IP Scanner** in the **EtherNet/IP Scanner** option (**EtherNet/IP -> EtherNet/IP Scanner**). Click **Add Device** to proceed and then press **Close** to close the device dialog.



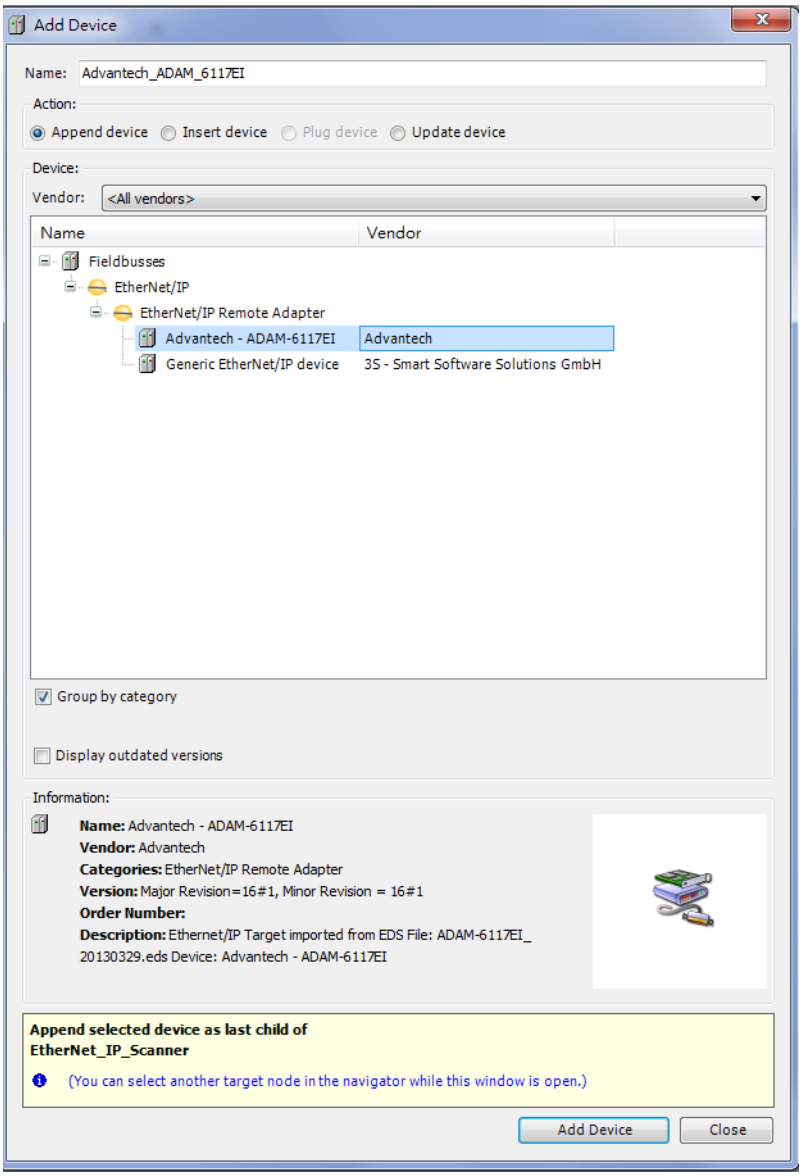
Now, you'll see EtherNet_IP_Scanner  in the device tree. Double-click the EtherNet_IP_Scanner icon to set configuration.

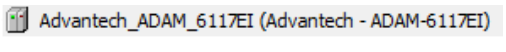


Right click on EtherNet_IP_Scanner 

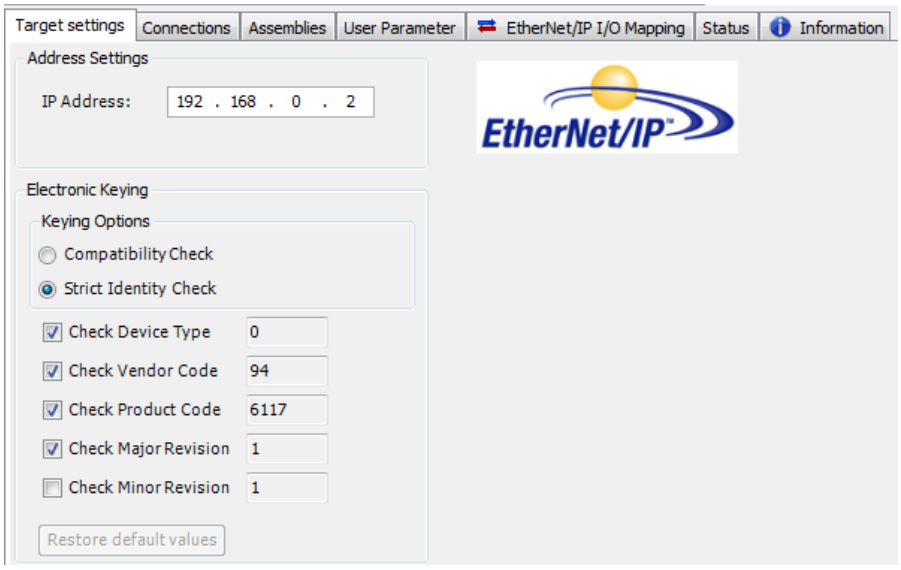
in the device tree and click **Add Device.** Choose the specified EtherNet/IP slave you want to add in the **EtherNet/IP Remote Adapter** option (**EtherNet/IP -> EtherNet/IP Remote Adapter**). Click **Add Device** to proceed and then press **Close** to close the device dialog.

Take the following as an example adding ADAM-6117EI into.

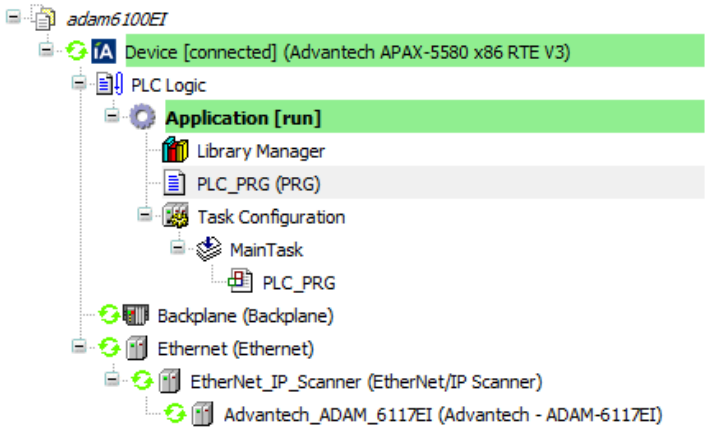


Now, you'll see ADVANTECH_ADAM_6117EI 

in the device tree. Double-click the ADVANTECH_ADAM_6117EI icon to set configuration including IP address,..etc.



After completely finish device configuration, it is necessary to login again.



6.4. Profinet

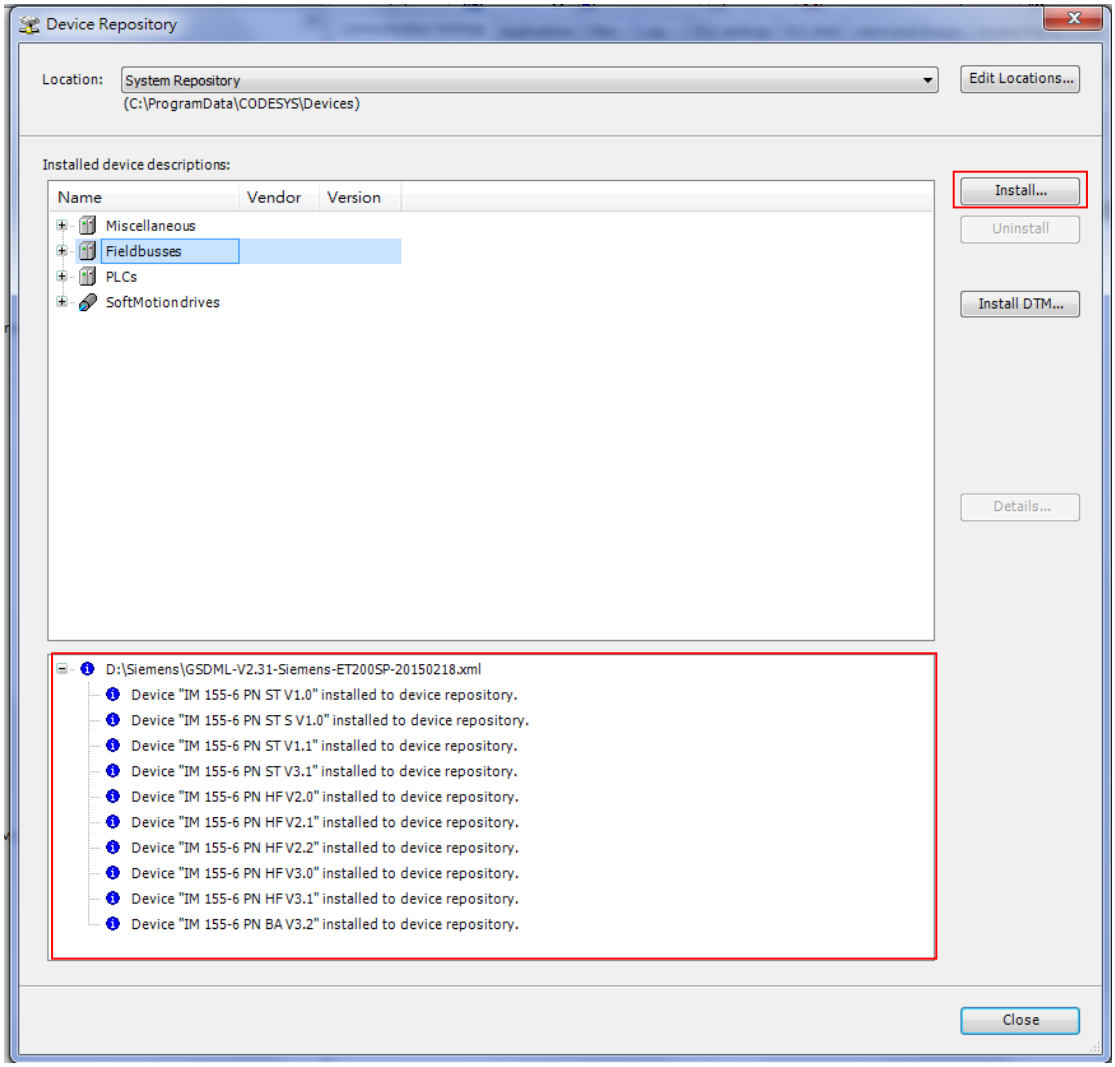
6.4.1. Profinet Client

6.4.1.2. Configuration Files Installation

Before connecting to slaves by using Profinet client, please install the related Profinet IO configuration files (GSDML*.xml) first.

Start CODESYS and perform command **Device Repository**  from the menu (**Tools -> Device Repository**). Click “Install” and then select the related GSD file you want to install.

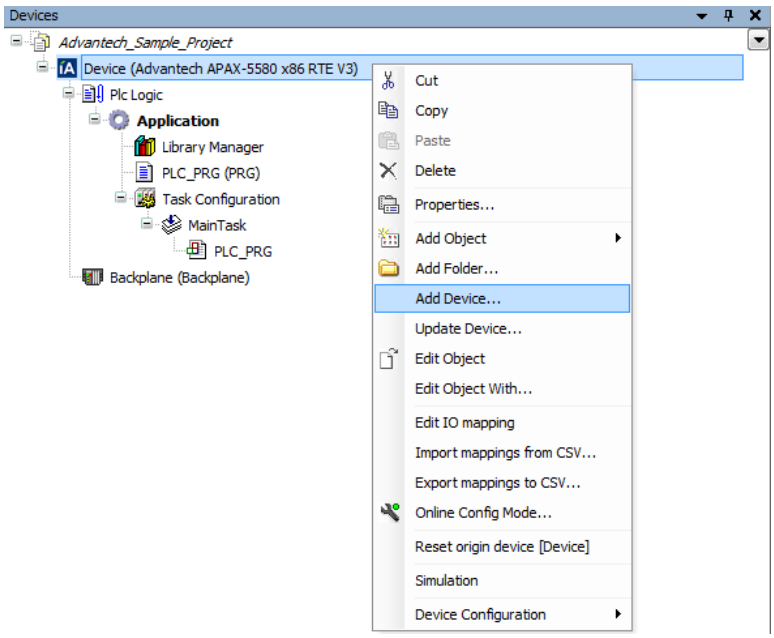
After installed successfully, the following information will be shown: “Device xxxx installed to device repository”.



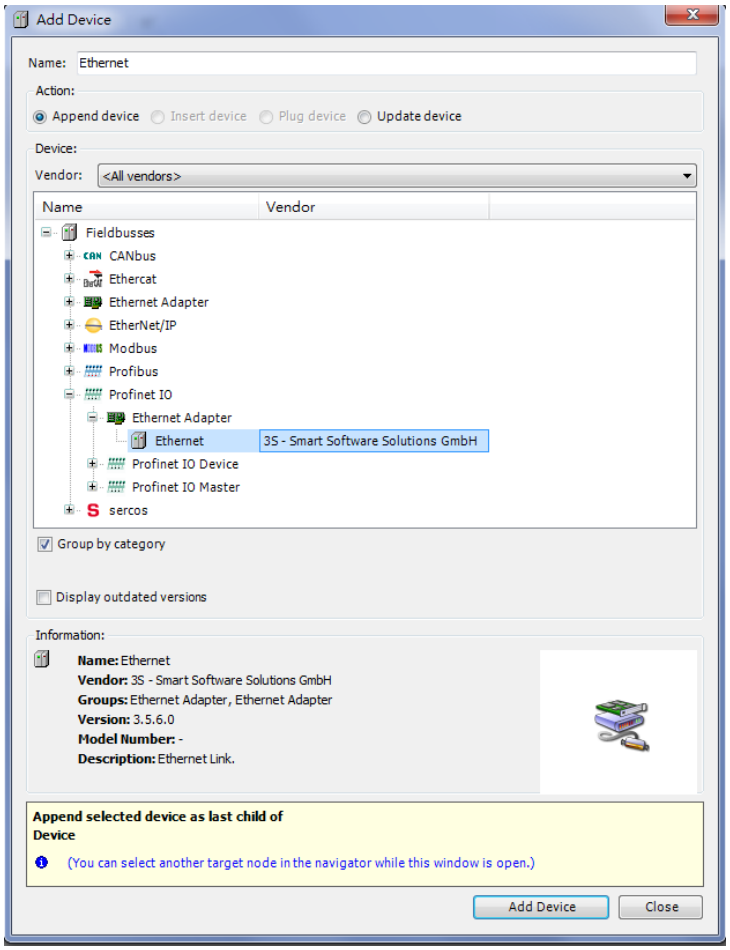
6.4.1.3. Scan for Slaves

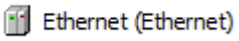
First, connect to your specified Advantech X86 RTE platform. Please refer to Chapter 3.4.

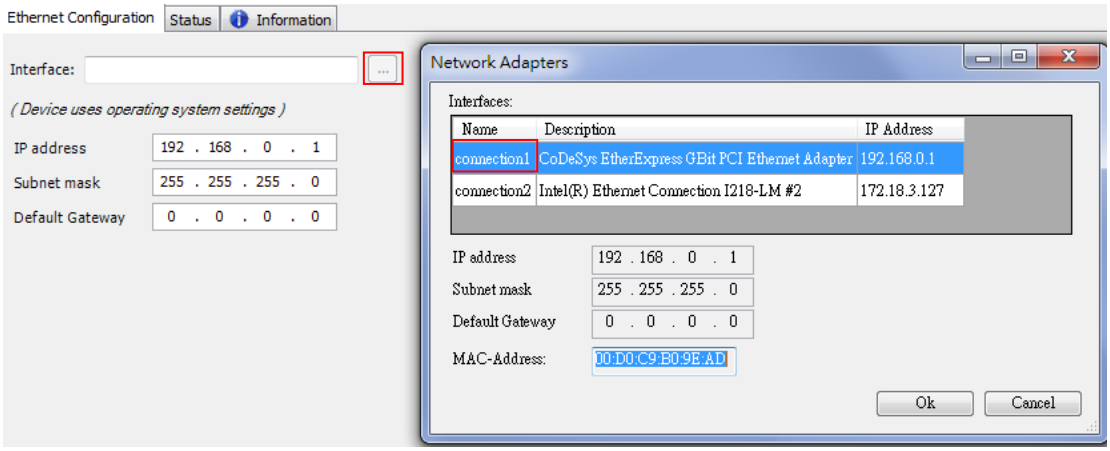
In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog.



Choose **Ethernet** in the **Ethernet Adapter** option (**Profinet IO -> Ethernet Adapter**) and click **Add Device** to proceed and then press Close to close the device dialog.



Now, you'll see Ethernet  in the device tree. Double-click the Ethernet icon to set configuration. Click **Interface** and then select **CODESYS Ethernet Adapter** (LAN#2)**.**

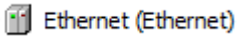


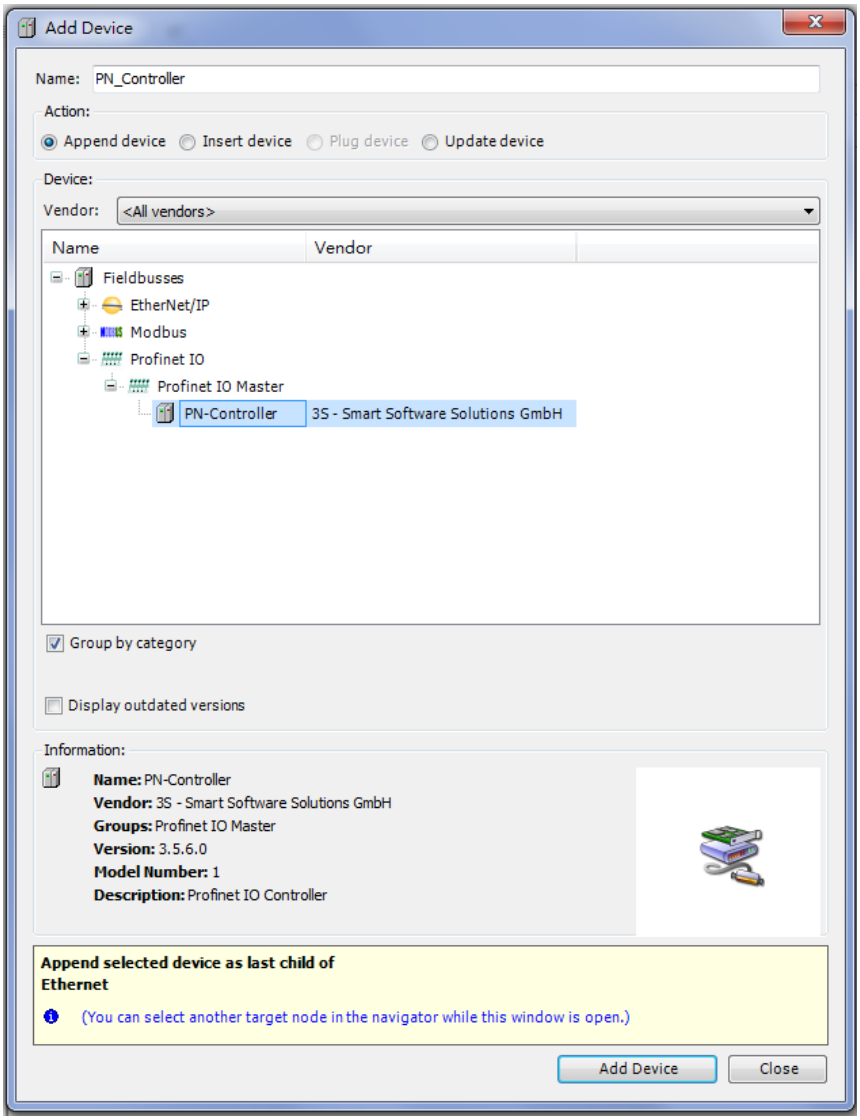
Note!

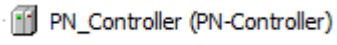
(1) Only LAN#2 can be used for Profinet.

(2) The IP address should be setup completely by using "Change Adapter settings" in Advantech X86 RTE platform.

(3) The Ethernet Adapter must not have a Unicode String Name. Please rename your adapter using ASCII characters only.

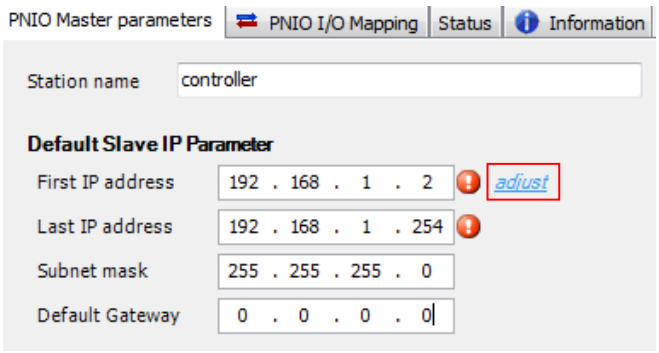
Right click on Ethernet  in the device tree and click **Add Device.** Choose **PN-Controller** in the **Profinet IO Master** option (**Profinet IO-> Profinet IO Master**). Click **Add Device** to proceed and then press **Close** to close the device dialog.



Now, you'll see PN-Controller 

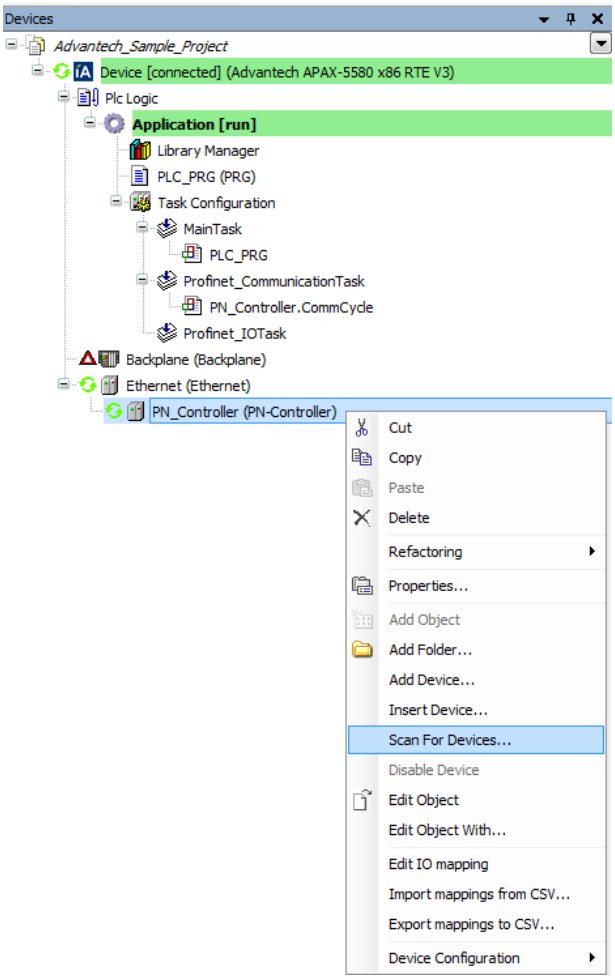
in the device tree. Double-click the PN-Controller icon to set configuration.

If default slave IP address parameter is invalid, please click "adjust" to automatically adjust to the right First and Last IP address. And make sure the subnet mask and gateway are all configured correctly.

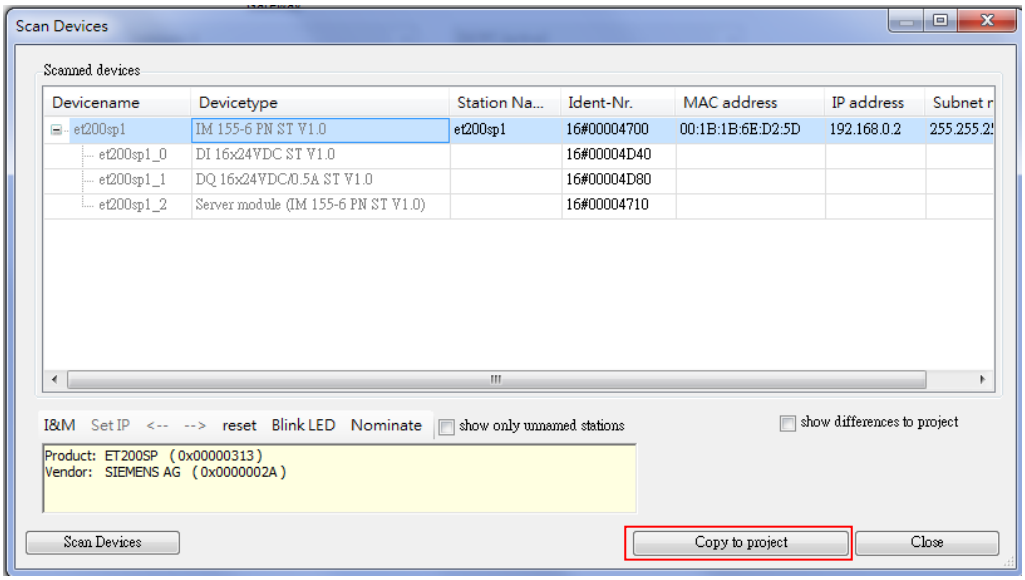


At the first scan, at least **once a login** must have been done. Otherwise, the Advantech X86 RTE platform must be running before a scan.

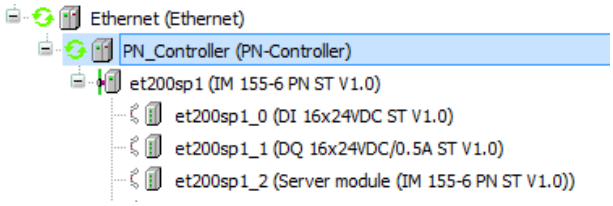
Choose the **PN-Controller** and right click **Scan For Devices** in context menu.



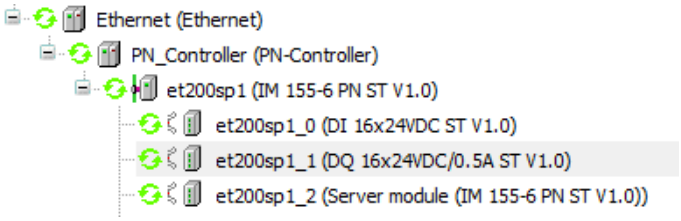
A list of all devices and modules are found during the last scan. Select the specified one device and then copy to the project. Or copy all listed devices to the project.



Now, you'll see the devices copied to the project under the PN-Controller.



After completely finish device configuration, it is necessary to login again.



6.5. EtherCAT

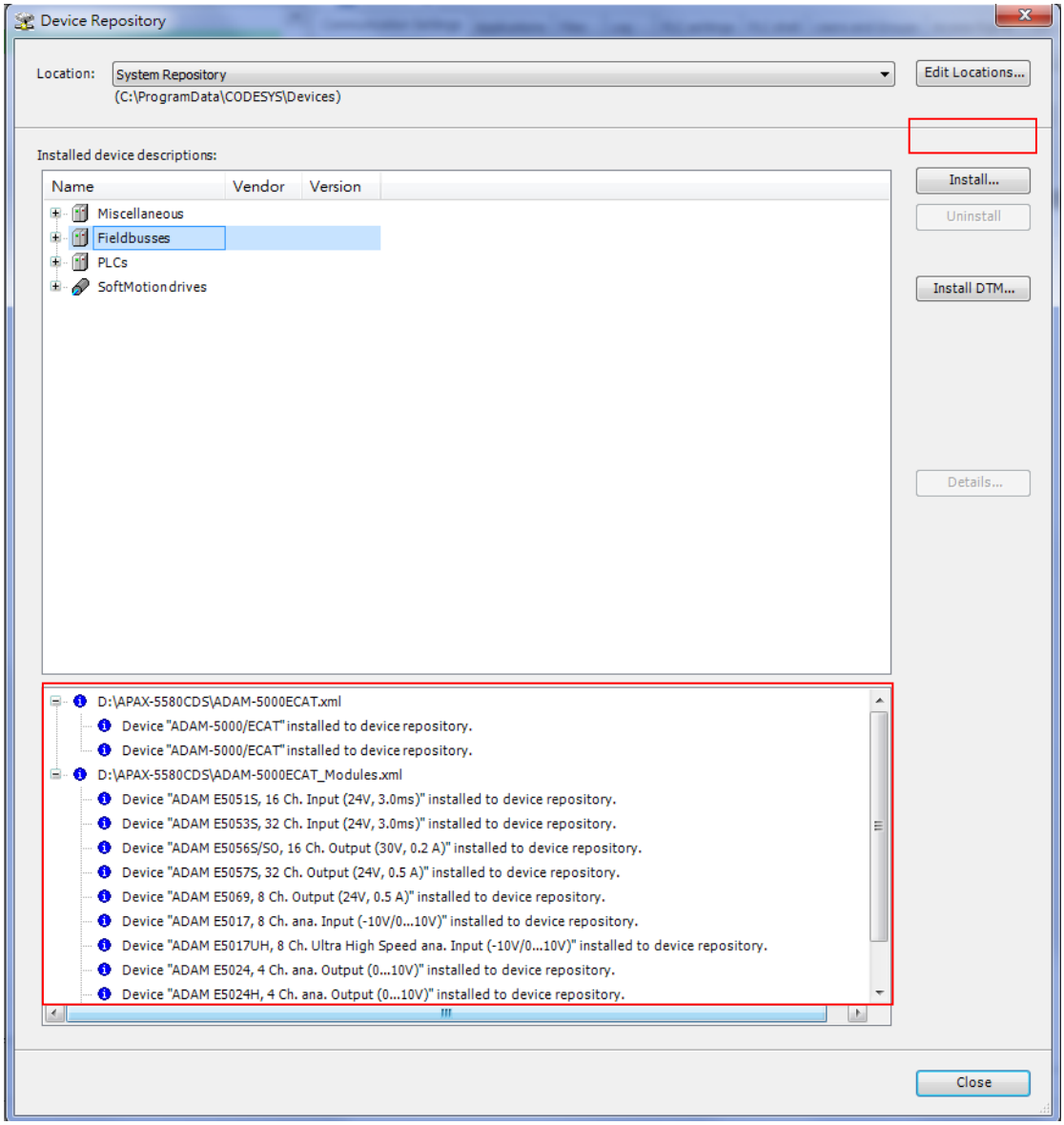
6.5.1. EtherCAT Client

6.5.1.2. Configuration Files Installation

Before connecting to slaves by using EtherCAT client, please install the related EtherCAT XML device description configuration files (*.xml) first.

Start CODESYS and perform command **Device Repository**  from the menu (**Tools -> Device Repository**). Click “Install” and then select the related XML file you want to install.

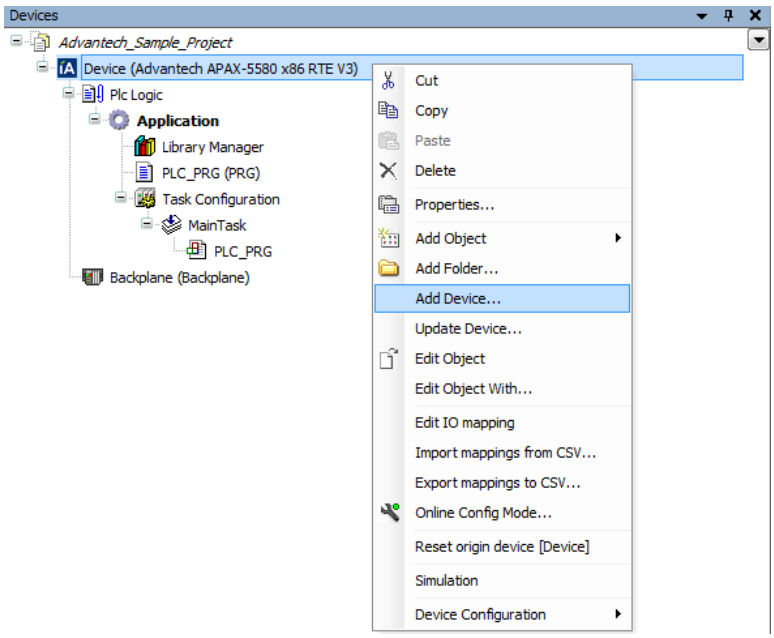
After installed successfully, the following information will be shown: “Device xxxx installed to device repository”.



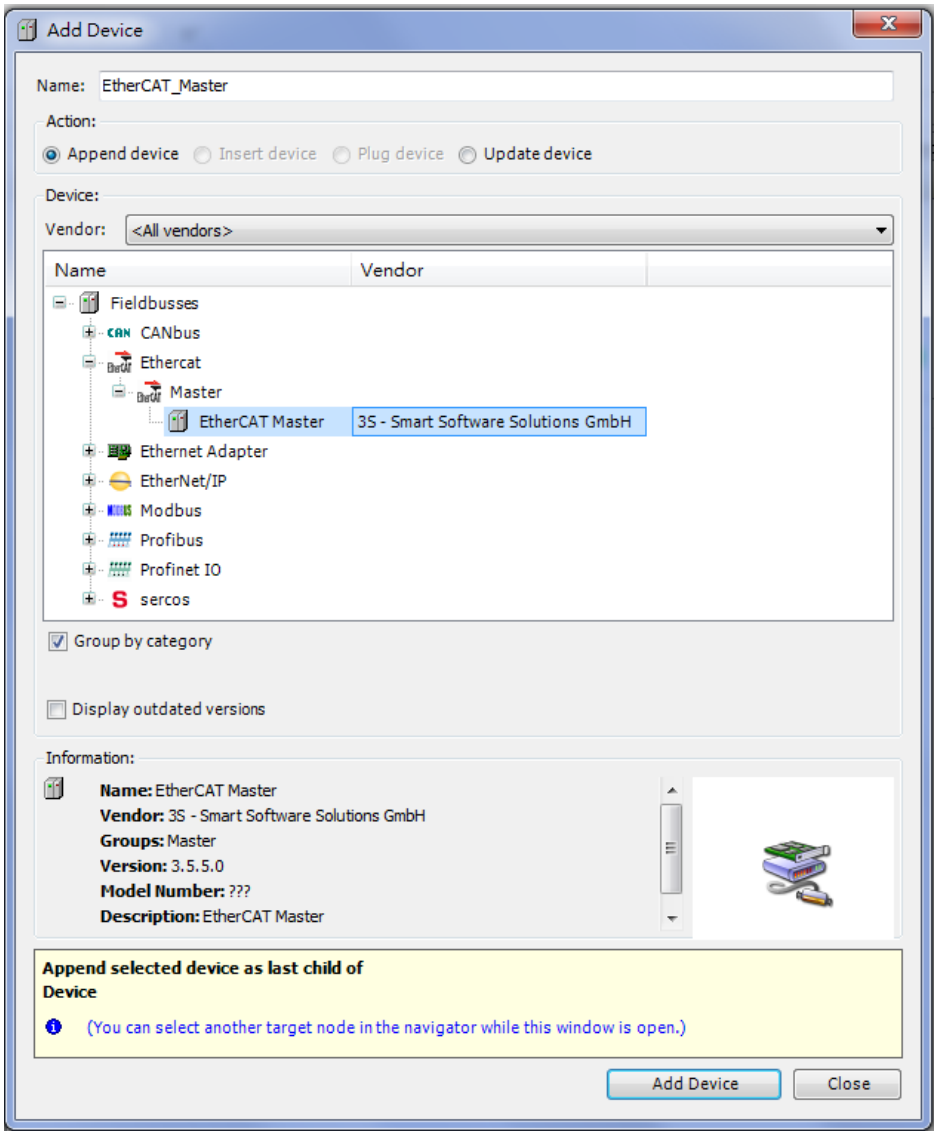
6.5.1.3. Scan for Slaves

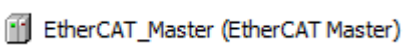
First, connect to your specified Advantech X86 RTE platform. Please refer to Chapter 3.4.

In Device Window, Right-click on **Device** and click **Add Device.** You will then be prompted for the Add Device dialog.



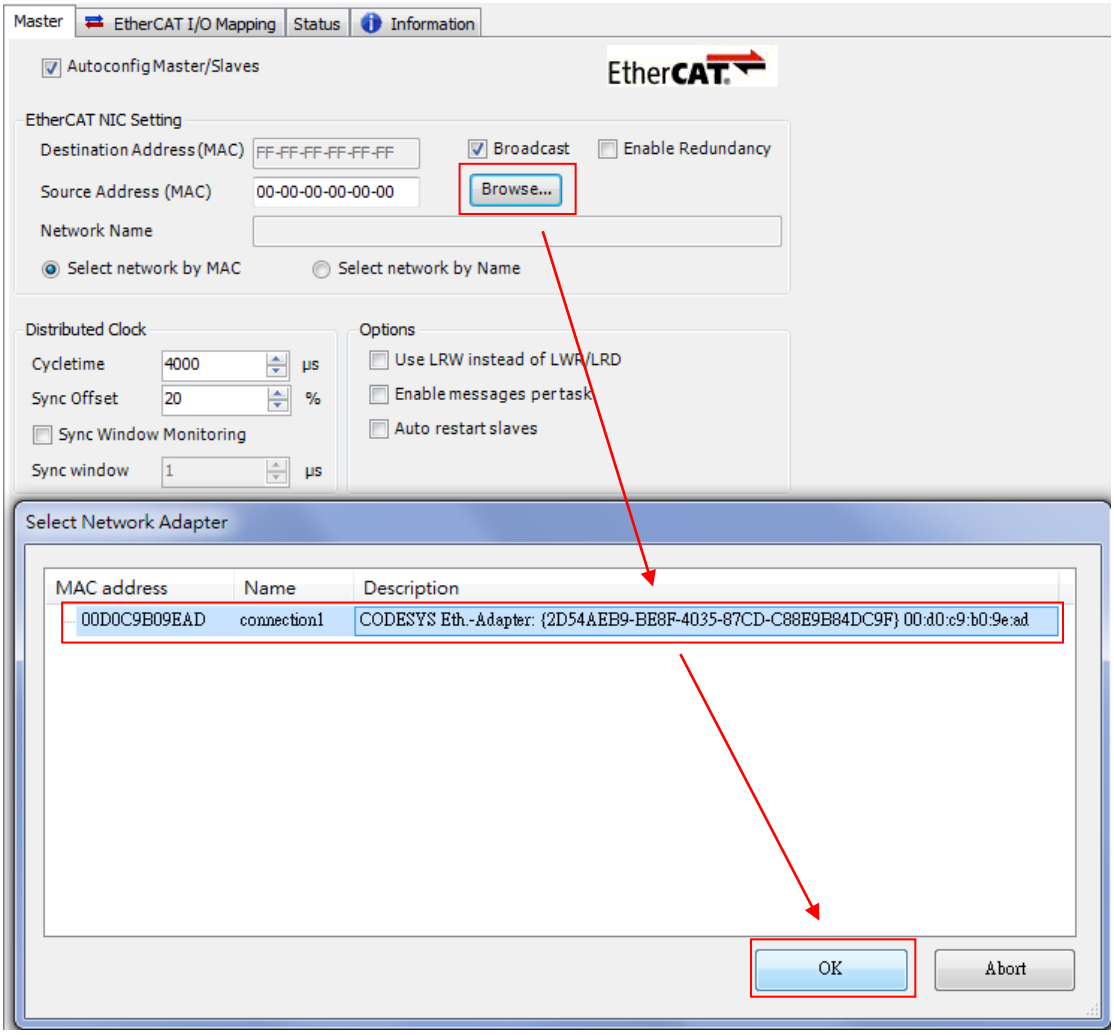
Choose **EtherCAT Master** in the **EtherCAT/Master** option and click **Add Device** to proceed and then press Close to close the device dialog.



Now, you'll see EtherCAT Master 

in the device tree.

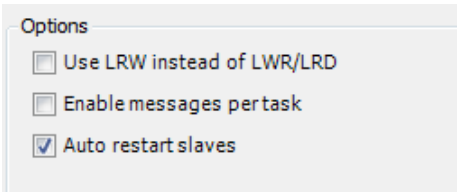
Double-click the EtherCAT Master icon to set configuration. Click **Browse** and then select **CODESYS Ethernet Adapter** (LAN#2)**.**



Note!

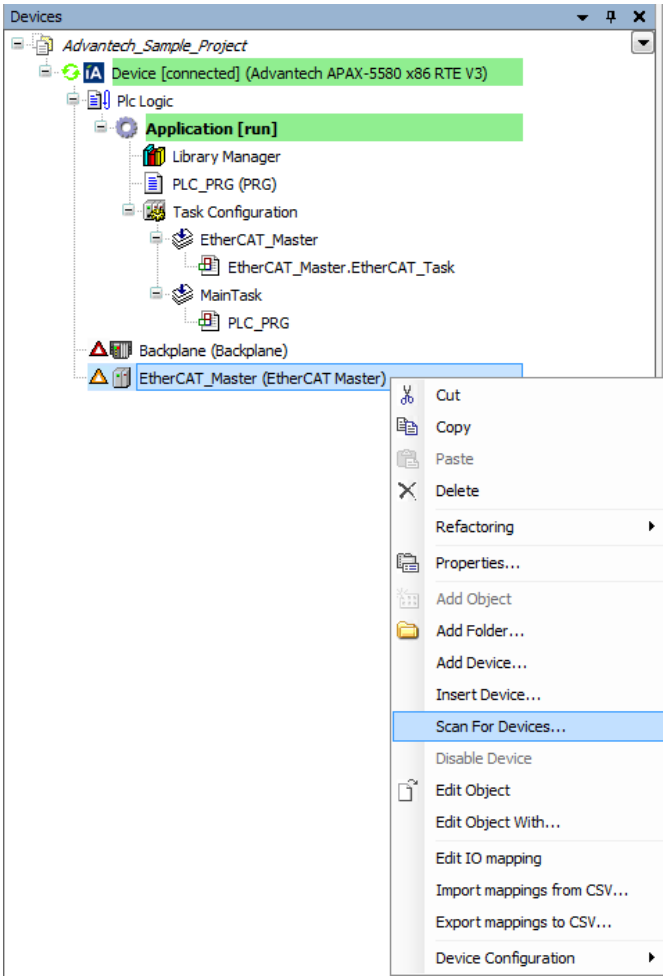
(4) Only LAN#2 can be used for EtherCAT.

(5) “Auto restart slaves” is suggested to be clicked.

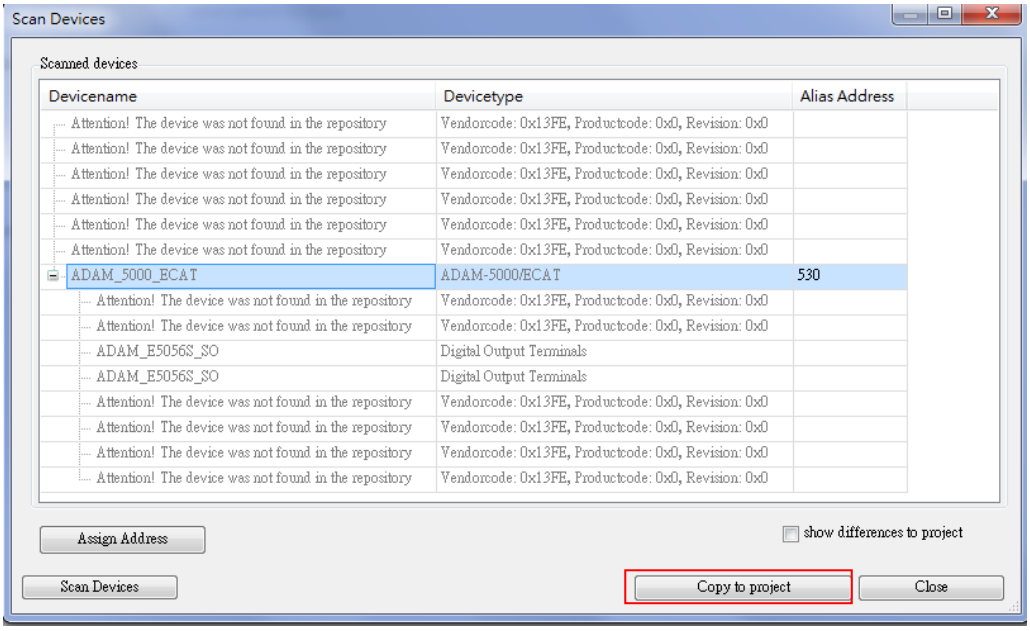


At the first scan, at least **once a login (and running)** must have been done. Otherwise, the Advantech X86 RTE platform must be running before a scan.

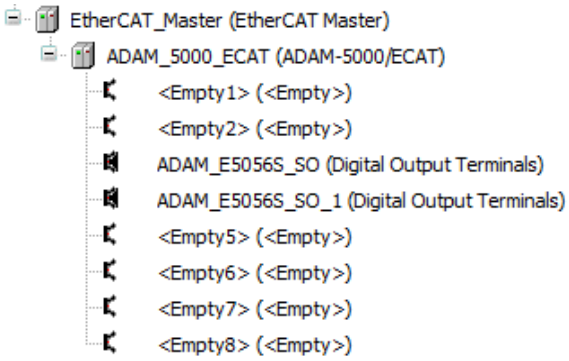
Choose the **EtherCAT_Master** and right click **Scan For Devices** in context menu.



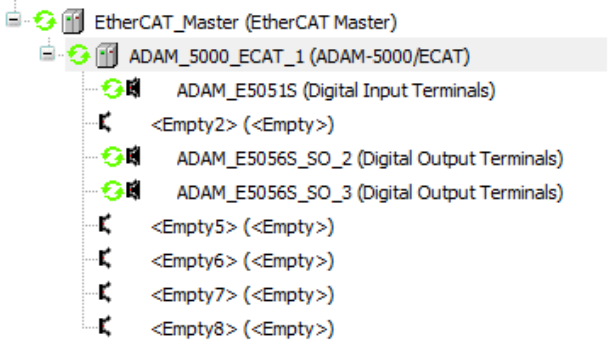
A list of all devices and modules are found during the last scan. Select the specified one device and then copy to the project. Or copy all listed devices to the project.



Now, you'll see the devices copied to the project under the **EtherCAT_Master**.



After completely finish device configuration, it is necessary to login again.



Chapter 7

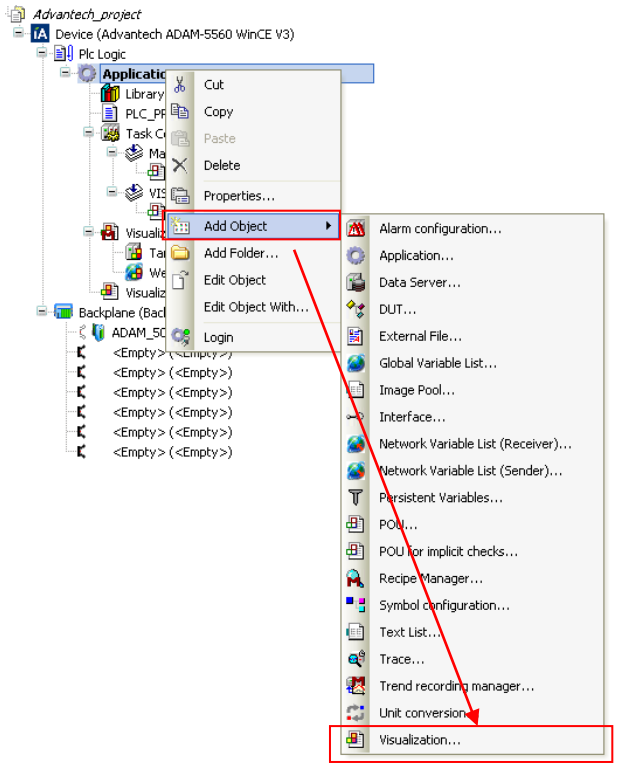
7. Examples

7.1. Visualization

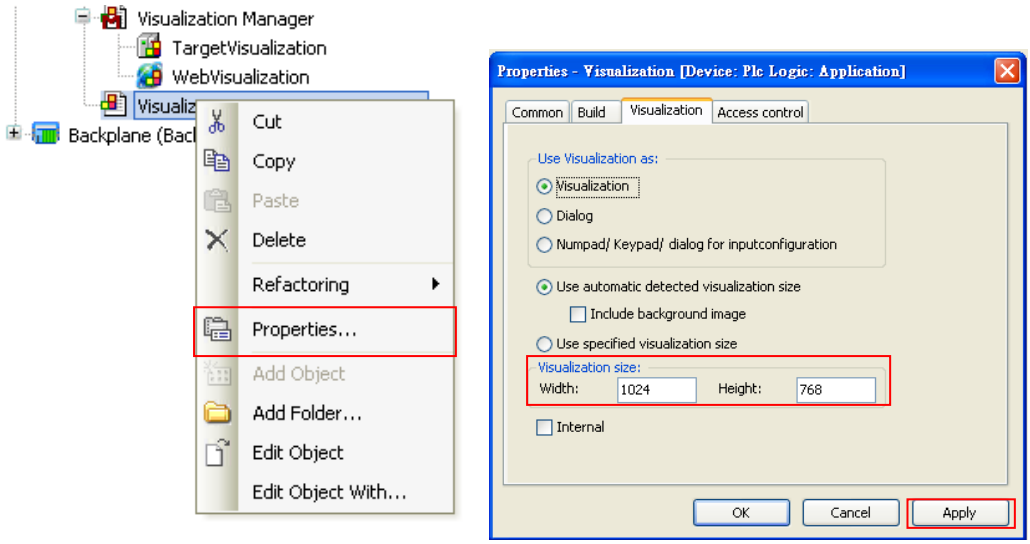
The CODESYS visualization is a graphical representation of the project variables which allows inputs to the program in online mode via mouse and keypad. The CODESYS visualization editor, which is part of the programming system, provides graphic elements which can be arranged as desired and can be connected with project variables. The following example project shows how to write a scrolling LED program in visualizations.

7.1.1. Create a new Visualization

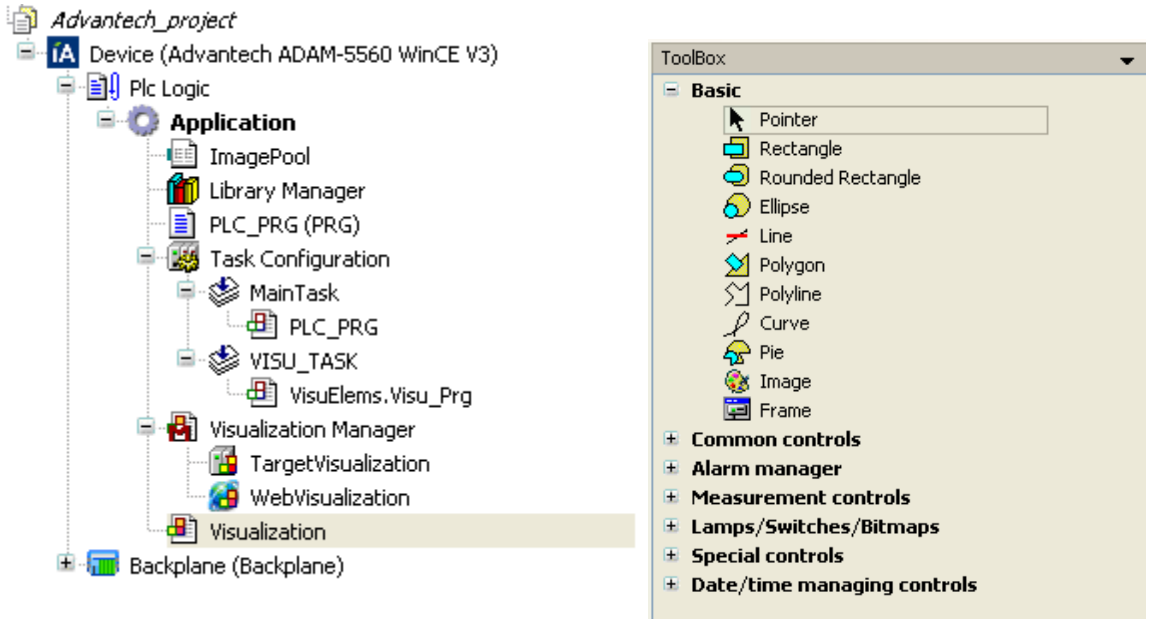
Step 1: To create a new visualization, right-click on **Application** and use command **Visualization** from the menu. You will then be prompted for **New visualization dialog.** Enter the name of the new visualization.



Step 2: Resize the width and height of the visualization object (number of pixels) by right-click on the visualization object.

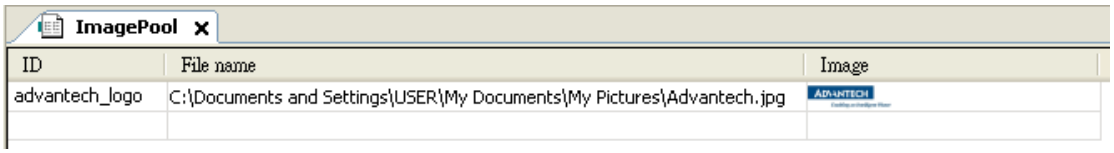


Step 3: Now, double-click on the visualization object and you'll see **ToolBox** on the right side. You can insert various geometric forms, as well as bitmaps, metafiles, buttons into your visualization by pushing down selected element into editor window.

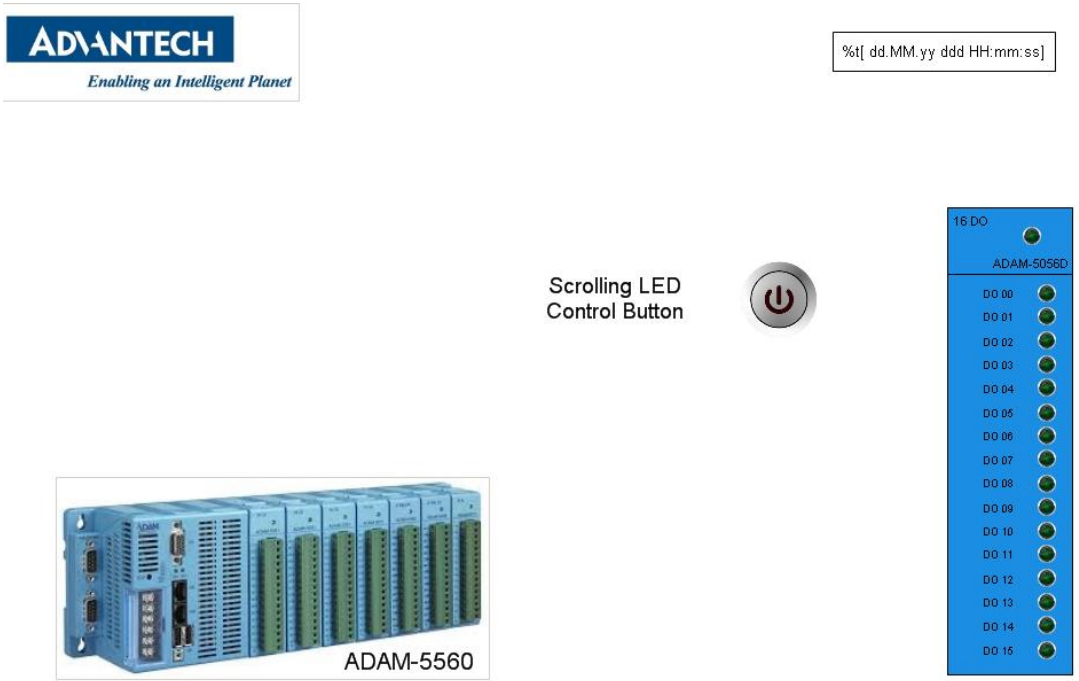


7.1.2. Visualize the Scrolling LED

Step 1: In this case, we need to insert Advantech images into the visualization, so we create an image pool. Again, right-click on **Application** and use command **Image Pool** from the menu. Enter string ID and the path of the image file



In order to visualize the scrolling LED, we insert lamps, rectangle, button and images.



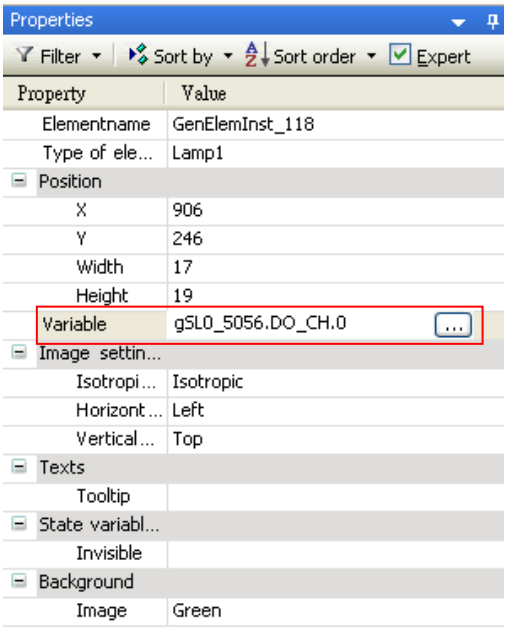
Step 2: Create a PLC program in PLC_PRG(PRG). For more detailed information about how to write a program, please refer to Chapter3.3.

```
PROGRAM PLC_PRG
VAR
    n: WORD := 1;
    vLED: WORD := 1;
    switch: BOOL;
    SL0_5056_STATUS: BOOL;
END_VAR
```

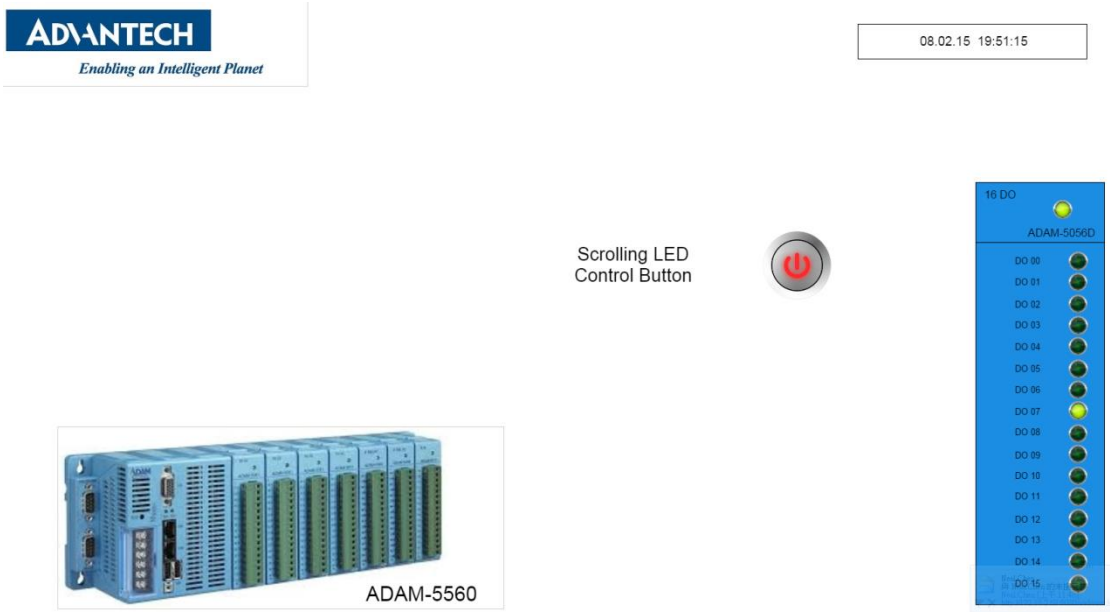
```
IF switch THEN
    IF n <= 16 THEN
        vLED:=vLED * WORD#2;
        n:=n+1;
    ELSE
        n:=1;
        vLED:=WORD#1;
    END_IF;
END_IF;
IF gSL0_5056.Err <> 0 THEN
    SL0_5056_STATUS := FALSE;
ELSE
    SL0_5056_STATUS := TRUE;
END_IF

gSL0_5056.DO_CH := vLED;
```

Step 3: Map variables to the lamp objects. For the first lamp, map to first bit of DO channel variable (**gSL0_5056.DO_CH**). For more detailed information about how to map variable, please refer to Chapter4.3.



Step 4: You have to connect to target device and running the program. Click on the button and the result was shown below.



7.2. Remnant Variables

Remnant variables retain their value throughout the usual program run time. They are declared as "Retain Variables" or "Persistent Variables". For keeping variables values even after the controller had been terminated or after the application has been reloaded, CODESYS offers different types:

7.2.1. Retain Variables

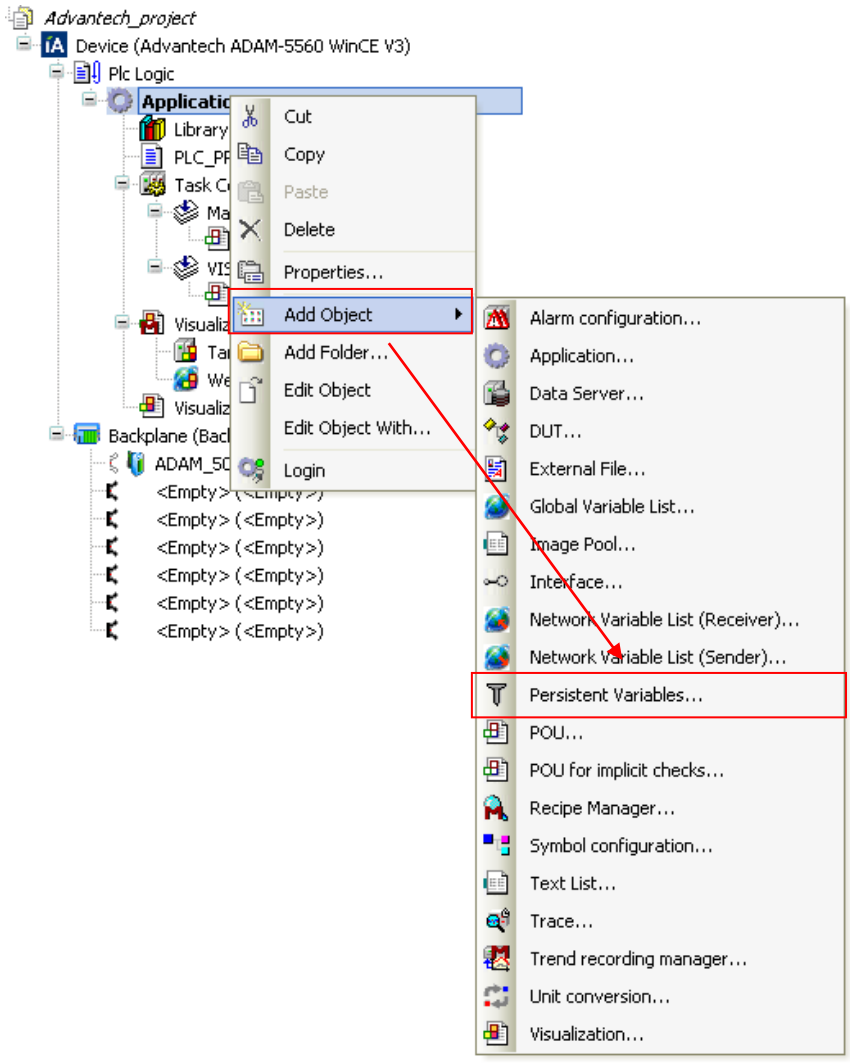
As we discuss how to declare variable in Chapter3.3, we can declare retain variables by adding the keyword RETAIN" in the declaration part, behind the keyword for the base variable's type.

Here, we declare 1 retain variable “var3_retain” with initial value 1000 and 2 normal variables.

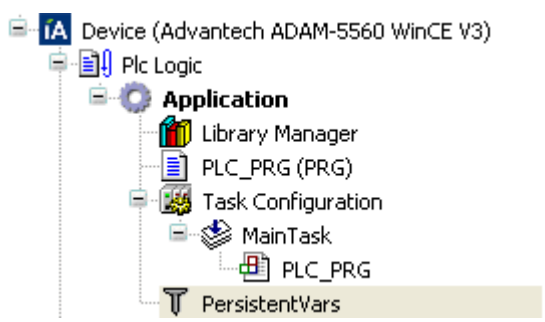
```
PLC_PRG
1  PROGRAM PLC_PRG
2  VAR
3      var1: int := 0;
4      var2_init: int := 1000;
5  END_VAR
6  VAR RETAIN
7      var3_retain: int := 1000;
8  END_VAR
```

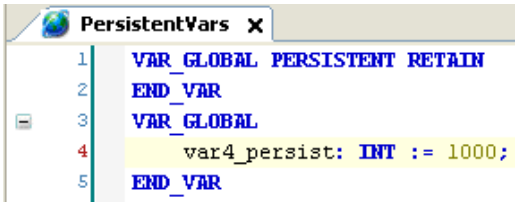
7.2.2. Persistent Variables

To create a new persistent variable, right-click on **Application** and use command **Persistent Variables** from the menu.



Double-click the object in the device tree. Here, we declare a global variable “var4_persist” with initial value 1000.





7.2.3. Variable Behavior

We want to investigate and compare the variable behavior between retain and persistent, so we keep adding their value in the body part of the PLC_PRG editor.

```
1  var1 := var1 +1;
2  var2_init := var2_init +1;
3  var3_retain := var3_retain +1;
4
5  var4_persist := var4_persist +1;
```

The result was shown below after we run our program on the target device.

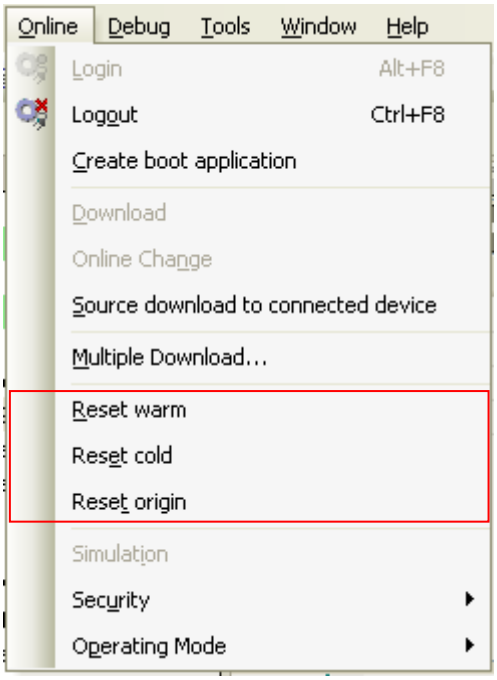
```
1  var1 37 := var1 37 +1;
2  var2_init 1037 := var2_init 1037 +1;
3  var3_retain 1037 := var3_retain 1037 +1;
4
5  var4_persist 1037 := var4_persist 1037 +1;RETURN
```

There three **Online Commands** for controlling the application program on a real or on the simulation target system after having logged in, including **Reset warm**, **Reset cold** and **Reset origin**.

The command **Reset warm** will reset (with exception of the retain and persistent variables) all variables of the currently active application to their initialization values. The situation is that which occurs in the event of a power outage or by turning the controller off, then on (warm restart) while the program is running.

The command **Reset cold** will reset (with exception of the persistent variables) all variables of the currently active application to their initialization values. The situation is that which occurs at the start of a program which has been downloaded just before to the target device.

The command **Reset origin** resets all variables of the currently active application, including the retain and persistent variables to their initialization values and erases the application on the target device.



The following is the overview on the behavior of remanent variables:

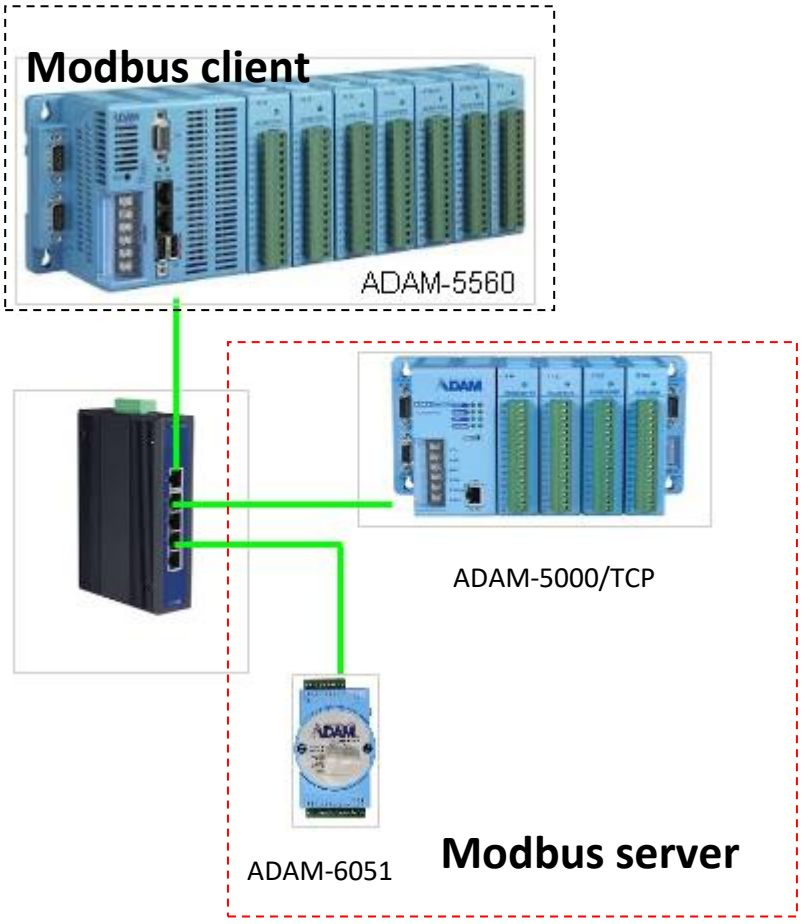
o = Value is maintained - = Value is initialized

After online command	**VAR**	**VAR RETAIN**	**VAR PERSISTENT RETAIN**
Reset warm <application>	-	o	o
Reset cold <application>	-	-	o
Reset origin <application>	-	-	-
Download <application>	-	-	o
Online Change <application>	o	o	o
Reboot the target device	-	o	o

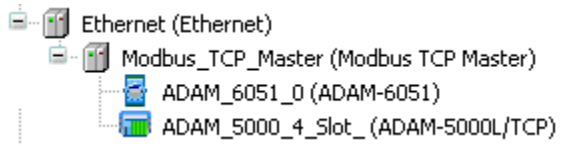
7.3. Modbus TCP Client

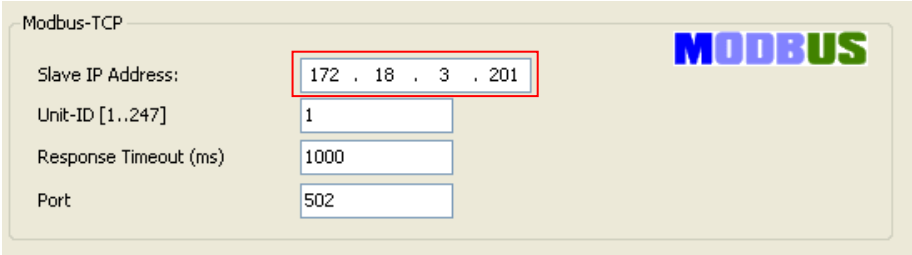
We use the following Advantech ADAM-5000 devices to demonstrate Modbus TCP client.

- ADAM-5560
- ADAM-5000/TCP with:
  – Slot 0: ADAM-5051(16-ch Digital Input)
  – Slot 1: ADAM-5056(16-ch Digital Output)
  – Slot 2: ADAM-5017(8-ch Analog Input)
  – Slot 3: ADAM-5024(4-ch Analog Output)
- ADAM-6051(14-ch Digital I/O with 2-ch Counter)



Step 1: Refer to Chapter5.1.2 and add Modbus TCP master to the project. Remember to enter the device's IP address.

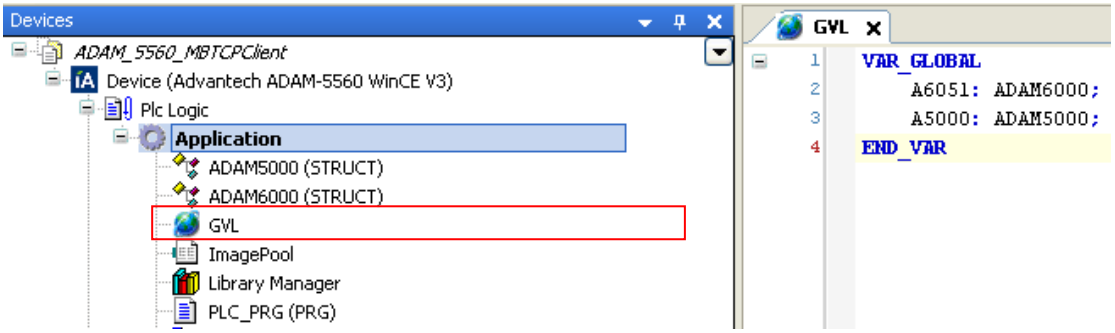




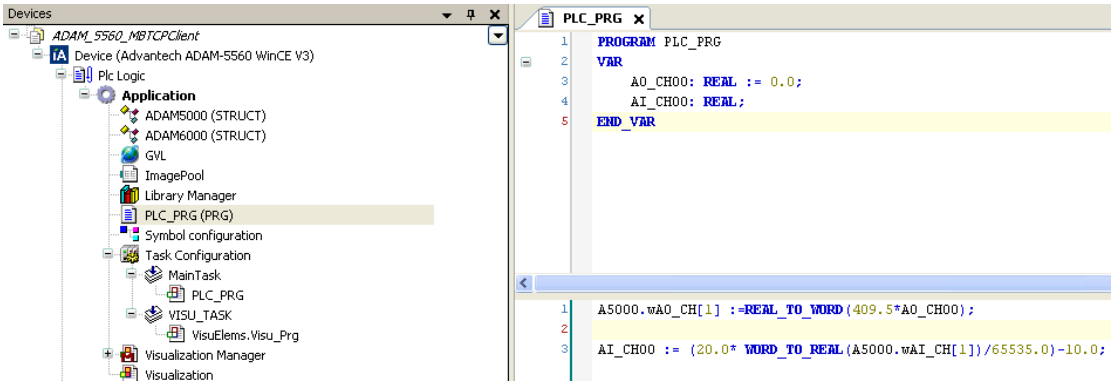
Step 2: Start to write our program. Define new structure to store ADAM-5000 and ADAM-6000 channel data and declare them as global variables.

```
TYPE ADAM5000 :
STRUCT
    byDI_CH : ARRAY [1..8] OF BYTE;
    byDO_CH : ARRAY [1..8] OF BYTE;
    wAI_CH:   ARRAY [1..32] OF WORD;
    wAO_CH:   ARRAY [1..32] OF WORD;
END_STRUCT
END_TYPE
```

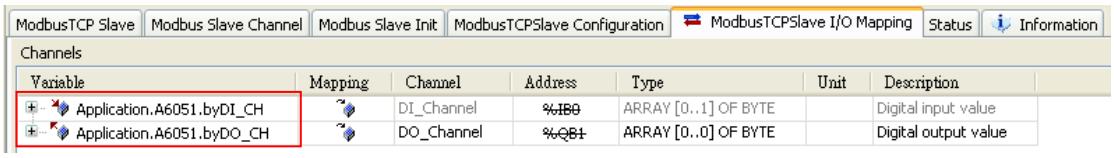
```
TYPE ADAM6000 :
STRUCT
    byDI_CH : ARRAY [1..2] OF BYTE;
    byDO_CH : ARRAY [1..2] OF BYTE;
    byAI_CH : ARRAY [1..8] OF WORD;
    byAO_CH : ARRAY [1..4] OF WORD;
END_STRUCT
END_TYPE
```

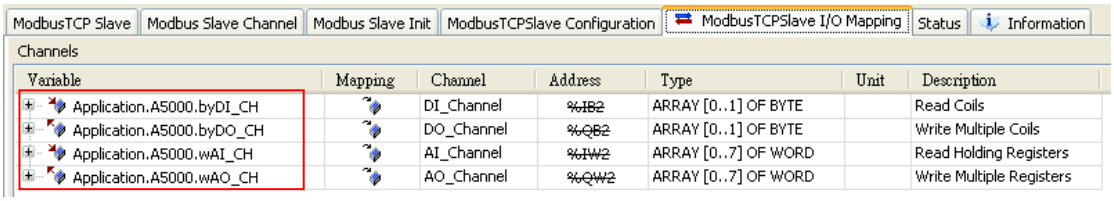


We write the program in PLC_PRG.



Step 3: Map Modbus data to the variables that we declared in previous step.





Step 4: Map all variables to the textfield objects in visualization.

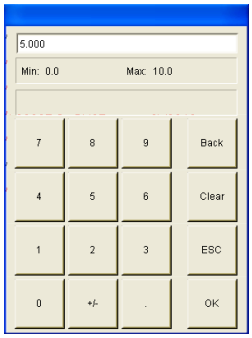
Object			Mapping variable
SL0_ADAM-5051DI_CH00	1X0001	%d	Text variables Text variable: A5000.byDI_CH[1].0 Tooltip variable:
SL1_ADAM-5056DO_CH00	0X0017	%d	Text variables Text variable: A5000.byDO_CH[1].0 Tooltip variable: Inputconfiguration OnDialogClosed: Configure... OnMouseClick: Configure... OnMouseDown: Configure... Toggle a V...: A5000.byDO_CH[1].0
SL2_ADAM-5017AI_CH00	3X0017	%1.3fV	Text variables Text variable: PLC_PRG.AI_CH00 Tooltip variable:

SL3_ADAM-5024AO_CH00	4X0025	%1.3fV	Text variables Text variable: PLC_PRG.AO_CH00 Tooltip variable Inputconfiguration OnDialogClosed: Configure... OnMouseClick: Configure... OnMouseDown: Configure... Write a Var...: Variable : , InputTyp...
ADAM-6051DI_CH00~07	1X0001~8	%d	Text variables Text variable: A6051.byDI_CH[1] Tooltip variable
ADAM-6051DO_CH00~01	0X0017~18	%d	Text variables Text variable: A6051.byDO_CH[1] Tooltip variable Inputconfiguration OnDialogClosed: Configure... OnMouseClick: Configure... OnMouseDown: Configure... Write a Var...: Variable : , InputType

Step 5: Compile our project and connect to the target device. The result was shown below.

For DO/AO channel, you can set its value by clicking on the textfield object and enter value in the pop-up dialog.

SL0_ADAM-5051DI_CH00	1X0001	0
SL0_ADAM-5051DI_CH01	1X0002	0
SL0_ADAM-5051DI_CH02	1X0003	0
SL0_ADAM-5051DI_CH03	1X0004	0
SL1_ADAM-5056DO_CH00	0X0017	0
SL1_ADAM-5056DO_CH01	0X0018	1
SL1_ADAM-5056DO_CH02	0X0019	1
SL1_ADAM-5056DO_CH03	0X0020	0
SL2_ADAM-5017AI_CH00	3X0017	0.001 V
SL3_ADAM-5024AO_CH00	4X0025	5.000 V
ADAM-6051DI_CH00~07	1X0001~8	255
ADAM-6051DI_CH08~11	1X0009~12	15
ADAM-6051DO_CH00~01	0X0017~18	0



7.4. Modbus TCP Server

We use the following Advantech ADAM-5000 devices to demonstrate Modbus TCP server.

- ADAM-5560

  –Slot 0: ADAM-5056(16-ch Digital Output)

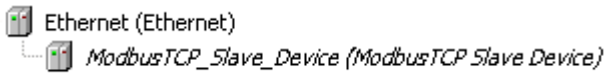
  – Slot 1: ADAM-5051(16-ch Digital Input)

  – Slot 2: ADAM-5017UH (8-ch Analog Input)

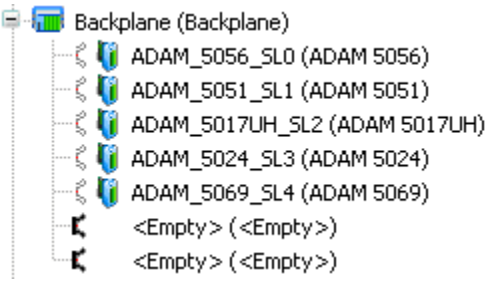
  – Slot 3: ADAM-5024(4-ch Analog Output)

  – Slot 4: ADAM-5069(8-ch Power Relay Output)

Step 1: Refer to Chapter5.1.3 and add the Modbus TCP slaves to the project. Rename them if necessary.



Step 2: Add the Advantech ADAM-5000 I/O modules to the project.

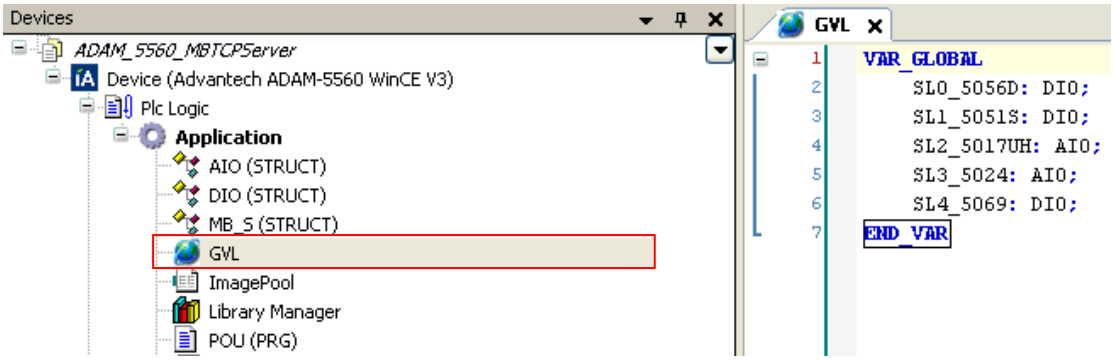


Step 3: Start to write our program. Define new structure to store DI/DO, AI/AO, Modbus channel data and declare them as global variables.

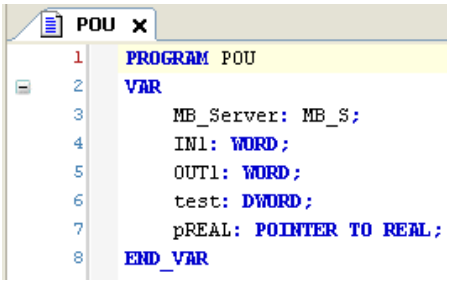
```
TYPE DIO :
STRUCT
    DI_CH:WORD;
    DO_CH:WORD;
    Err:WORD;
END_STRUCT
END_TYPE
```

```
TYPE AIO :
STRUCT
    CH00:REAL;
    CH01:REAL;
    CH02:REAL;
    CH03:REAL;
    CH04:REAL;
    CH05:REAL;
    CH06:REAL;
    CH07:REAL;
    Err:WORD;
END_STRUCT
END_TYPE
```

```
TYPE MB_S :
STRUCT
    MB3:ARRAY [1..100] OF WORD;
    MB4:ARRAY [1..100] OF WORD;
END_STRUCT
END_TYPE
```



We write the program in PLC_PRG.

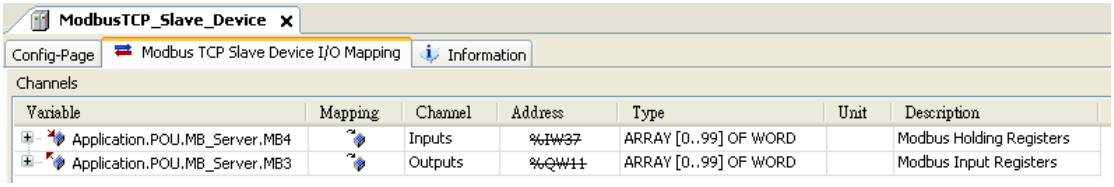



```
SL0_5056D.DO_CH:= MB_Server.MB4[1];//5056 DO CH0~15
pREAL:= ADR(MB_Server.MB4[3]);
SL3_5024.CH00 := pREAL^;//5024 AO CH0
pREAL:= ADR(MB_Server.MB4[5]);
SL3_5024.CH01 := pREAL^;//5024 AO CH1
pREAL:= ADR(MB_Server.MB4[7]);
SL3_5024.CH02 := pREAL^;//5024 AO CH2
pREAL:= ADR(MB_Server.MB4[9]);
SL3_5024.CH03 := pREAL^;//5024 AO CH3
SL4_5069.DO_CH:= MB_Server.MB4[11];//5069 CH0~7

MB_Server.MB3[1]:= SL0_5056D.Err ;//5056 Status
MB_Server.MB3[2]:= SL1_5051S.Err ;//5051 Status
MB_Server.MB3[3]:= SL2_5017UH.Err;//5017UH Status
MB_Server.MB3[4]:= SL3_5024.Err;//5024 Status
MB_Server.MB3[5]:= SL4_5069.Err;//5069 Status

MB_Server.MB3[11]:= SL1_5051S.DI_CH;//5051 CH0~15
MB_Server.MB3[13]:= %IW18;//5017UH CH0
MB_Server.MB3[14]:= %IW19;
MB_Server.MB3[15]:= %IW20;//5017UH CH1
MB_Server.MB3[16]:= %IW21;
MB_Server.MB3[17]:= %IW22;//5017UH CH2
MB_Server.MB3[18]:= %IW23;
MB_Server.MB3[19]:= %IW24;//5017UH CH3
MB_Server.MB3[20]:= %IW25;
MB_Server.MB3[21]:= %IW26;//5017UH CH4
MB_Server.MB3[22]:= %IW27;
MB_Server.MB3[23]:= %IW28;//5017UH CH5
MB_Server.MB3[24]:= %IW29;
MB_Server.MB3[25]:= %IW30;//5017UH CH6
MB_Server.MB3[25]:= %IW31;
MB_Server.MB3[27]:= %IW32;//5017UH CH7
MB_Server.MB3[28]:= %IW33;
```

Step 4: Set Modbus TCP server configuration and map Modbus data to the variables that we declared in previous step.



Step 5: Map all variables to the textfield objects in visualization.

Object			Mapping variable
ADAM-5056 CH0~CH15	4X0001	%d	Text variables Text variable: POU.MB_Server.MB4[1] Tooltip variable:
ADAM-5024 CH0	4X0003	%1.3f mA	Text variables Text variable: SL3_5024.CH00 Tooltip variable:
ADAM-5069 CH0~CH7	4X0011	%d	Text variables Text variable: POU.MB_Server.MB4[11] Tooltip variable:
ADAM-5051 CH0~CH15	3X0011	%d	Text variables Text variable: POU.MB_Server.MB3[11] Tooltip variable:
ADAM-5017UH CH0	3X0013	%1.3f mA	Text variables Text variable: SL2_5017UH.CH00 Tooltip variable:

Step 6: Compile our project and connect to the target device. The result was shown below.

ADVANTECH
Enabling an Intelligent Planet

26.02.15 Thu 01:03:07

Module Name	Address	Status
ADAM-5056	3X0001	0
ADAM-5051	3X0002	0
ADAM-5017UH	3X0003	0
ADAM-5024	3X0004	0
ADAM-5069	3X0005	0

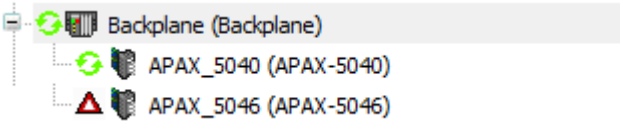
Module Channel	Address	Value
ADAM-5056 CH0~CH15	4X0001	0
ADAM-5024 CH0	4X0003	0.000 mA
ADAM-5024 CH1	4X0005	0.000 mA
ADAM-5024 CH2	4X0007	0.000 mA
ADAM-5024 CH3	4X0009	0.000 mA
ADAM-5069 CH0~CH7	4X0011	0
ADAM-5051 CH0~CH15	3X0011	0
ADAM-5017UH CH0	3X0013	4.000 mA
ADAM-5017UH CH1	3X0015	4.000 mA
ADAM-5017UH CH2	3X0017	4.000 mA
ADAM-5017UH CH3	3X0019	4.000 mA
ADAM-5017UH CH4	3X0021	4.000 mA
ADAM-5017UH CH5	3X0023	4.000 mA
ADAM-5017UH CH6	3X0025	4.000 mA
ADAM-5017UH CH7	3X0027	4.000 mA

Chapter 8

8. Diagnosis and Troubleshooting

8.1. Error Notification

In chapter 4, we introduce how to write a program to control Advantech I/O modules. If the Advantech modules are correctly configured, it will show a green circle icon next to the device name in the device tree after performing command **Login** and **Start**. If it shows a red triangle , it means that I/O module encountered several errors while running.



8.2. Log Information

We can get log information from **Advantech CODESYS** or **target machine**, i.e. ADAM-5560, Advantech X86 RTE platforms.

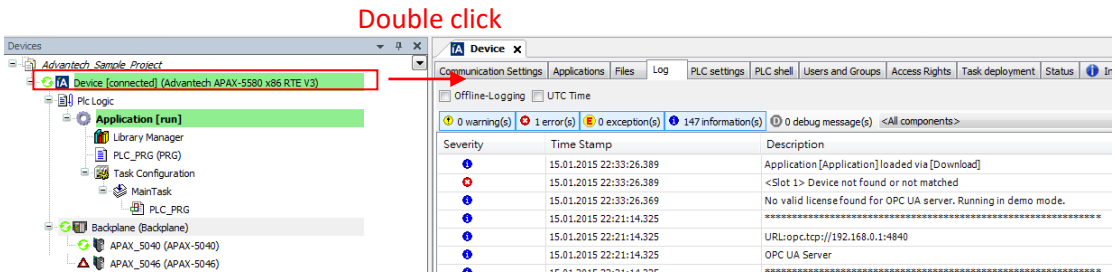
In Advantech CODESYS development environment, double click the device name in the device tree to open **Device editor**. Select the **Log** dialog and it will display the log of the Advantech I/O module. A log entry line contains the following information:

Severity: There are four categories: warnings, errors, exceptions, information. The display of the entries of each category can be switched on or off by using the corresponding button from the bar above the listing. Each button always contains the current number of loggings in the respective category.

Time Stamp: Date and Time.

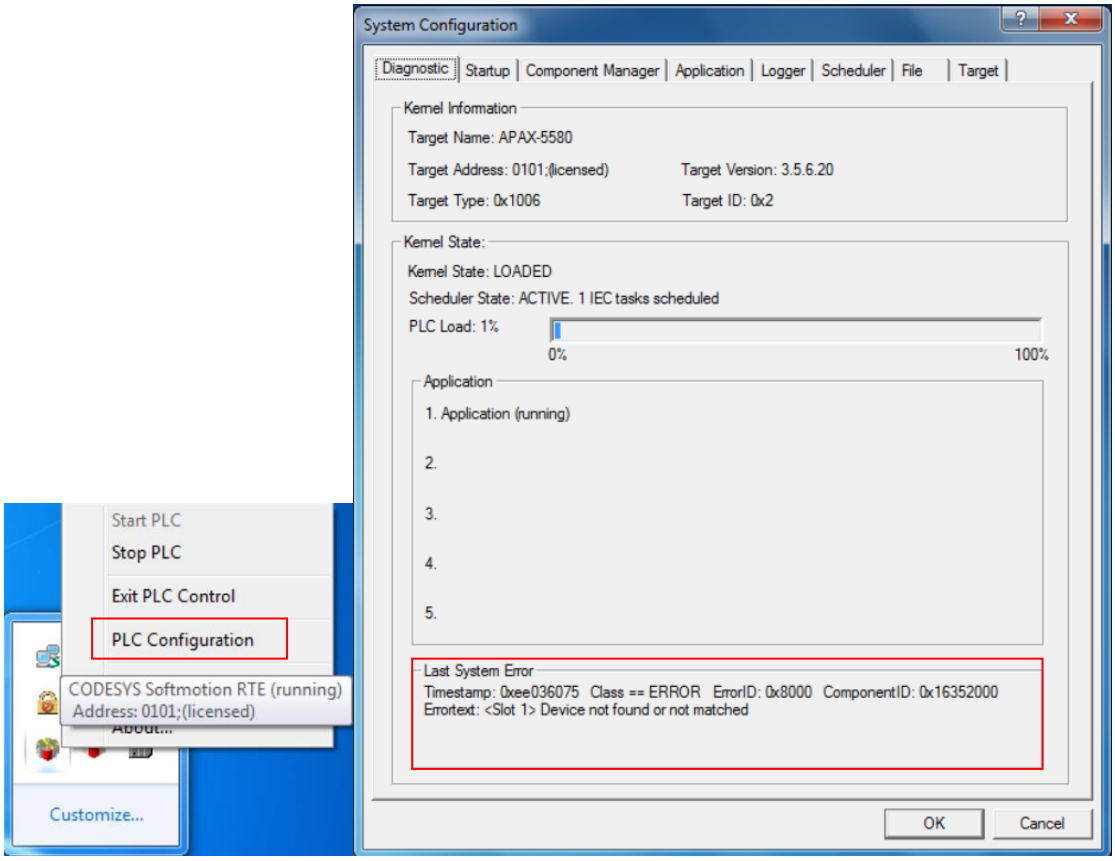
Description: Description of the event, for example “Device not found or not matched."

Component: ID and name of the component



On target machine, we can also get error ID from CODESYS RTE runtime.

In the Advantech X86 RTE platform's environment, open runtime by right-clicking the runtime icon  which is available on the lower-right corner of the desktop. Click **PLC configuration** and then it will show the system error in the diagnostic of system configuration.



8.3. Error ID

Following table is the error ID for I/O modules.

Error ID	Description
0x8000	The module didn't exist or match the setting module. Make sure that the setting module matches for the device that is being plugged and check your module is plugged in target device appropriately.
0x8001	The system failed to open the module. Please close all programs and reboot. If the system cannot returns to normal condition or the error occurred, please contact Advantech for technical support.
0x8002	The system was unable to complete configuration. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8003	The system failed to read value from the module. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8004	The system failed to write value to the module. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8005	For counter module, the system failed to start/stop counter. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.

0x8006	For counter module, the system failed to clear counting value. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8007	For counter module, the system failed to clear overflow flag. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.
0x8008	For counter module, the system failed to clear alarm flag. Please power-off the system and plug the module again. If the error occurred, please replace a new module and contact Advantech for technical support.